(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－147800
（P2001－147800A）
(43)公開日　平成13年５月29日(2001.5.29)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　7/24		Ｇ０６Ｆ　7/24	Ａ	
12/00	５６０	12/00	５６０Ｆ	
12/06	５３０	12/06	５３０Ａ	
15/16	６２０	15/16	６２０Ａ	
17/30		15/40	３１０Ｂ	

審査請求　未請求　請求項の数14　ＯＬ　（全 34 頁）　最終頁に続く

(21)出願番号　特願平11－330997

(22)出願日　平成11年11月22日(1999.11.22)

(71)出願人　599074648
ターボデータラボラトリー有限会社
東京都台東区松が谷１－９－12　ＳＰＫビルディング604号
(72)発明者　古庄　晋二
神奈川県横浜市神奈川区松見町４丁目1101番地７コートハウス菊名804号
(74)代理人　100103632
弁理士　窪田　英一郎　（外１名）

(54)【発明の名称】　情報処理システム、並びに、この情報処理システムを利用したソート方法、コンパイル方法およびジョイン方法

(57)【要約】
【課題】　著しく高速に、かつ、安定した処理時間で、配列のソートなどを実現する。
【解決手段】　分散メモリ型の情報処理装置において、提示メモリモジュール14－1、14－3が自己のメモリモジュール内でソートされた要素を順位番号とともに、スイッチ３０などにより分割されたバス２４を介して、判定メモリモジュール14－2、14－4に与える。判定メモリモジュールは、与えられた順位番号に基づき、受理した要素の順位番号の候補を示す仮想順位番号を算出して、当該仮想順位番号を、他のバス２４を介して、提示メモリモジュールに返送する。提示メモリモジュールは、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位番号を更新する。

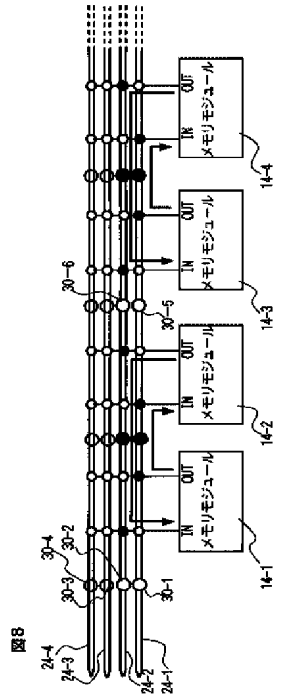


図8

1

【特許請求の範囲】

【請求項１】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムであって、

前記メモリモジュールのＭＰＵが、自己の把握する配列の部分を構成する要素のソートを実行して、前記要素を特定の順序にしたがって並べ替えるソート手段と、

前記自己の把握する前記部分が配列中に占める位置にしたがって、前記ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達し、或いは、所定のバスを介して、他のメモリモジュールからの前記要素および順位番号を受理するＩ／Ｏと、

前記要素および順位番号を受理した場合に、自己の把握する要素との比較により、受理した要素の順位番号の候補である仮想順位番号を算出して、前記他のメモリモジュールに返送する順位番号算出手段と、

前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位を確定する順位確定手段とを備え、

前記要素および順位番号を送出する側のメモリモジュールである提示メモリモジュールと、前記要素および順位番号受理して仮想順位番号を算出する側のメモリモジュールである判定メモリモジュールとの通信により、前記配列の要素の順位番号を確定することを特徴とする情報処理システム。

【請求項２】　前記メモリモジュールが、

確定した順位番号にしたがって、処理対象となる要素を特定して何れかのバスに送出する要素特定／送出手段と、

前回の処理対象となった要素と送出された要素とを比較する要素比較手段と、

同一の要素が送出された場合には、その値をカウントアップする、同一の要素の存在数を示す同一値個数カウンタとを備え、前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて何れかの送出するように構成され、さらに、

何れかのメモリモジュールが、

送出された前回の処理対象となった要素および関連するカウンタの値を受理して、これらを関連付け、かつ、受理した順序で配置する配列を備えたことを特徴とする請求項１に記載の情報処理システム。

2

【請求項３】　前記メモリモジュールが、

前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、その値をカウントアップする、重複のない順位番号を示す値番号カウンタと、

送出された要素に関して、前回の処理対象となった要素と送出された要素とが同一の場合には値番号カウンタの値を、重複のない当該要素の順位番号と決定し、その一方、これらが異なる場合には、カウントアップされた値番号カウンタの値を、重複のない当該要素の順位番号と決定して、当該順位番号を更新する順位番号更新手段とを備えたことを特徴とする請求項２に記載の情報処理装置。

【請求項４】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した配列のソート方法であって、（ａ）メモリモジュールにおいて、自己が把握する配列の部分を構成する要素をソートするステップと、（ｂ）前記自己の把握する前記部分が配列中に占める位置にしたがって、前記配列の部分を把握するメモリモジュールのうち、要素および順位番号を送出する側の提示メモリモジュール、および、要素および順位番号を受理する側の判定メモリモジュールを決定するステップと、（ｃ）提示メモリモジュールにおいて、ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達するステップと、（ｄ）判定メモリモジュールにおいて、所定のバスを介して他のメモリモジュールからの前記要素および順位番号を受理するステップと、（ｅ）前記判定メモリモジュールにおいて、当該判定メモリモジュールが把握する要素の順位番号に基づき、受理した要素の順位番号の候補を示す仮想順位番号を算出して、当該仮想順位番号を、前記提示メモリモジュールに返送するステップと、（ｆ）前記提示メモリモジュールにおいて、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位番号を更新するステップと、（ｇ）前記ステップ（ｄ）～（ｆ）が終了するごとに、当該ステップ（ｄ）～（ｆ）により所定の順位番号が付された要素に関する提示メモリモジュールと判定メモリモジュールからなるメモリモジュール群を、それぞれ、提示メモリモジュール群、および、判定メモリモジュール群の一方として、ステップ（ｄ）～（ｆ）を繰り返すことにより、各メモリモジュール群における要素の順位番号を更新することにより、配列の各要素の順位番

号を確定することを特徴とするソート方法。

【請求項５】　前記ステップ（ｅ）が、（e1）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号とに基づき、仮想順位番号を算出するステップを含むことを特徴とする請求項４に記載のソート方法。

【請求項６】　前記ステップ（ｆ）が、（f1）受理した仮想順位番号を、ステップ（ｃ）にて送出した要素の順位番号に代入するステップを含むことを特徴とする請求項４または５に記載のソート方法。

【請求項７】　さらに、（ｈ）提示メモリモジュール群において、当該提示メモリモジュール群を構成するメモリモジュールにて把握されている要素が、当該メモリモジュール群においていくつ存在しているかを示す重複数を算出するステップを備え、

前記ステップ（ｃ）が、（c1）同一の要素を重複して伝達しないように、ソートされた要素を、その順位番号および重複数とともに、他のメモリモジュールに伝達するステップを含み、前記ステップ（ｅ）が、（e2）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号および重複数とに基づき、仮想順位番号を算出するステップを含み、かつ、

前記ステップ（ｆ）が、（f2）仮想順位番号と、ステップ（ｃ）における要素の送出時の順位番号との差に基づき、当該要素と同一の要素の順位番号を決定するステップを含むことを特徴とする請求項４に記載のソート方法。

【請求項８】　初期的に提示メモリモジュールが単独のメモリモジュールであり、かつ、受信モジュールも単独のメモリモジュールであり、

（ｄ）～（ｆ）のステップが終了する毎に、ｎ（ｎ：１以上の整数）がインクリメントされるような2^nのメモリモジュールからなる提示メモリモジュール群と、2^nのメモリモジュールからなる判定メモリモジュール群が形成されることを特徴とする請求項４ないし７の何れか一項に記載のソート方法。

【請求項９】　請求項４ないし６の何れか一項に記載された方法により、配列をソートし、かつ、当該ソートされた配列に基づき、前記配列中の要素が、重複なく、かつ、所定の順序にて配置された新たな配列を生成するコンパイル方法であって、（ｉ）所定のメモリモジュールにおいて、順位番号にしたがって処理対象となる要素を送出するステップと、（ｊ）前回の処理対象となった要素と同一の要素が送出された場合には、同一の要素の存在数を示す同一値個数カウンタをカウントアップし、その一方、前回の処理対象となった要素と異なる要素が送出された場合には、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて、これらを送出するステップと、（ｋ）前回の処理対象となった要素、および、関連する同一値カウンタの値を受理して、これらを関連付けて新たな配列中に配置するステップとを備え、（ｌ）ステップ（ｉ）～（ｊ）を繰り返すことにより、前記新たな配列中に、要素およびその存在数が関連付けられて配置されることを特徴とするコンパイル方法。

【請求項１０】　さらに、（ｍ）何れかのモジュールにおいて、ステップ（ｊ）にて送出される要素および関連する同一値個数カウンタの値をモニターするステップを備え、

当該何れかのモジュールにより、ステップ（ｋ）が実行されることを特徴とする請求項９に記載のコンパイル方法。

【請求項１１】　（ｎ）当該配列の要素を把握するメモリモジュールにおいて、処理対象となっている要素の順位番号および当該要素の存在数をそれぞれ格納する順位番号カウンタおよび同一値個数カウンタを設けるとともに、および、前回の処理対象となった要素を一時的に格納するレジスタを設けるステップと、（ｏ）順位番号にしたがって、当該順位番号が付された要素を把握するメモリモジュールにおいて、当該要素を第１のバスに送出するステップと、（ｐ）配列の要素を把握するメモリモジュールにおいて、受理した要素とレジスタの内容とを比較して、これらが一致する場合には、存在数をカウントアップする一方、これらが一致しない場合には、レジスタの内容および存在数カウンタの値を、第２のバスに送出した後に、レジスタの内容および存在数カウンタの値を更新するステップと、（ｑ）何れかのメモリモジュールにおいて、前記レジスタの内容および存在数カウンタの値を、それぞれ要素および当該要素の存在数として、配列中に配置するステップとを備えたことを特徴とする請求項９に記載のコンパイル方法。

【請求項１２】　ステップ（ｎ）が、さらに、（n1）処理対象となっている要素に関して、重複のない順位番号を格納する値カウンタを設けるステップを含み、

前記ステップ（ｐ）が、（p1）受理した要素とレジスタの内容とを比較して、これらが一致する場合に、当該処理対象となる要素の順位番号に、値番号カウンタの値を付与する一方、これらが一致しない場合に、値番号カウンタをカウントアップし、処理対象となる要素の順位番号に、カウントアップされた値番号カウンタの値を付与するステップを含むことを特徴とする請求項１１に記載のコンパイル方法。

【請求項１３】　請求項４ないし請求項８の何れか一項に記載のソート方法、および、請求項９ないし１２の何れか一項に記載のコンパイル方法を用いて、複数の配列の共有化を実現する配列のジョイン方法であって、

（ｒ）複数の配列を合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行す

5

るステップと、（ｓ）前記合併した配列中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな配列を生成するステップとを備えたことを特徴とする配列のジョイン方法。

【請求項１４】　ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した複数の配列のジョイン方法であって、請求項４ないし請求項８の何れか一項に記載のソート方法、および、請求項９ないし１２の何れか一項に記載のコンパイル方法を用いて、複数の配列の共有化を実現する配列のジョイン方法において、

前記メモリモジュールが、それぞれ、レコード番号に基づき、要素を格納した配列である値リストにおける所定の要素を指定するために、レコード番号に対応する位置に、値リストを示すポインタ値を配置したポインタ配列を備え、（ｒ1）複数の値リストを合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行するステップと、（ｔ）前記合併した値リスト中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな値リストを生成するとともに、前記要素の順位番号を、重複した要素の存在しない場合の当該要素の順位番号に更新するステップと、（ｕ）前記重複した要素の存在しない場合の要素の順位番号からなる配列を、新たな値リストを示すための、新たなポインタ配列とするステップとを備えたことを特徴とするジョイン方法。

【発明の詳細な説明】

【０００１】

【産業上の技術分野】本発明は、分散メモリ型の情報処理装置に関し、より詳細には、極めて高速にソート、コンパイルおよびジョインの処理を実現可能な情報処理装置に関する。

【０００２】

【従来の技術】社会全体のさまざまな場所にコンピュータが導入され、インターネットをはじめとするネットワークが浸透した今日では、そこここで、大規模なデータが蓄積されるようになった。このような大規模データを処理するには、膨大な計算が必要で、そのために並列処理を導入しようと試みるのは自然である。

【０００３】さて、並列処理アーキテクチャは「共有メモリ型」と「分散メモリ型」に大別される。前者（「共有メモリ型」）は、複数のプロセッサが１つの巨大なメモリ空間を共有する方式である。この方式では、プロセッサ群と共有メモリ間のトラフィックがボトルネックとなるので、百を越えるプロセッサを用いて現実的なシステムを構築することは容易ではない。したがって、例えば１０億個の浮動小数点変数の平方根を計算する際、単一ＣＰＵに対する加速比は、せいぜい１００倍ということになる。経験的には、３０倍程度が上限である。

【０００４】後者（「分散メモリ型」）は、各プロセッサがそれぞれローカルなメモリを持ち、これらを結合してシステムを構築する。この方式では、数百～数万ものプロセッサを組み込んだハードウェアシステムの設計が可能である。したがって、上記１０億個の浮動小数点変数の平方根を計算する際の単一ＣＰＵに対する加速比を、数百～数万倍とすることが可能である。

【０００５】

【発明が解決しようとする課題】数百を越える多数のプロセッサによる並列処理の潜在的需要は大きいといわれているが、上述したように、現在の現実的なハードウェア技術でこれを実現しようとすると、分散メモリ型以外の手法による設計は困難である。分散メモリ型においては、個々のプロセッサに付属するメモリの容量が小さいため、並列処理の主たる目的の一つである大規模データ（通常は、配列）の保持および処理において、これを複数プロセッサおよびそれぞれに付属するメモリが分掌する必要がある。

【０００６】しかしながら、複数プロセッサおよびそれぞれに付属するメモリが配列を分掌する場合には、バス上のデータの衝突を防止するためのバス調停が困難であり、各プロセッサが並列的に作動できなければ、プロセッサの利用効率を向上できず、その結果、処理の高速化を図ることができないなどの問題点がある。そこで、本発明は、以下のように、種々の目的を達成する。

【０００７】（１）アルゴリズム的にバス上のデータの衝突が発生せず、バス調停が不要であり、これにより、バスのバンド幅をフルに生かして処理速度を向上させる。

（２）プロセッサ（好ましくは複数のプロセッサ）と目盛りとを備えたメモリモジュールを多数組み合わせて、これらによる並列処理を可能とし、それぞれのメモリモジュールを有効利用し、かつ、各メモリモジュール内のプロセッサに独立した処理を割り当てられるようにし、これにより、メモリモジュールの有効利用により処理速度をさらに向上させる。

（３）ソート対象のデータの大きさを「Ｎ」とした場合に、Ｏ（Ｎ）のデータの大きさしか必要としない。（従来のソート処理では、最悪の場合にＯ（Ｎ＊Ｎ）やＯ（Ｎ＊Log(Ｎ)）のデータ量を必要とし得る。）

（４）処理時間が安定しており、最悪の場合でも、予想可能な処理速度が保証される。つまり、本発明は、著し

く高速に、かつ、安定した処理時間で、配列のソートなどが可能な情報処理装置を提供することを目的とする。

【０００８】

【課題を解決するための手段】本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムであって、前記メモリモジュールのＭＰＵが、自己の把握する配列の部分を構成する要素のソートを実行して、前記要素を特定の順序にしたがって並べ替えるソート手段と、前記自己の把握する前記部分が配列中に占める位置にしたがって、前記ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達し、或いは、所定のバスを介して、他のメモリモジュールからの前記要素および順位番号を受理するＩ／Ｏと、前記要素および順位番号を受理した場合に、自己の把握する要素との比較により、受理した要素の順位番号の候補である仮想順位番号を算出して、前記他のメモリモジュールに返送する順位番号算出手段と、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位を確定する順位確定手段とを備え、前記要素および順位番号を送出する側の提示メモリモジュールと、前記要素および順位番号受理して仮想順位番号を算出する側の判定メモリモジュールとの通信により、前記配列の要素の順位番号を確定することを特徴とする情報処理システムにより達成される。

【０００９】本発明によれば、提示メモリモジュールによる要素および順位番号の提示があるバスを介して実行され、判定メモリモジュールにより仮想順位番号が算出され、当該仮想順位番号が、他のバスを介して、提示メモリモジュールに与えられる。したがって、提示メモリモジュールおよび判定メモリモジュールにおいて、並列的にソート処理を進めることができ、かつ、バスの衝突も回避することが可能となる。

【００１０】本発明の好ましい実施態様においては、前記メモリモジュールが、確定した順位番号にしたがって、処理対象となる要素を特定して何れかのバスに送出する要素特定／送出手段と、前回の処理対象となった要素と送出された要素とを比較する要素比較手段と、同一の要素が送出された場合には、その値をカウントアップする、同一の要素の存在数を示す同一値個数カウンタとを備え、前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて何れかの送出するように構成され、さらに、何れかのメモリモジュールが、送出された前回の処理対象となった要素および関連するカウンタの値を受理して、これらを関連付け、かつ、受理した順序で配置する配列を備えている。この実施態様によれば、何れかのメモリモジュールにおいて、要素およびその重複数が、所定の順序で受理され、これにより、重複のない要素の配列、および、各要素の存在数を示す配列を作成することが可能となる。すなわち、これにより、重複のない要素のリスト、および、もとの配列において各要素の存在する数を容易に把握することができる。

【００１１】本発明の別の実施態様によれば、前記メモリモジュールが、前記要素比較手段が、前回の処理対象となった要素と送出された要素とが異なると判断した場合に、その値をカウントアップする、重複のない順位番号を示す値番号カウンタと、送出された要素に関して、前回の処理対象となった要素と送出された要素とが同一の場合には値番号カウンタの値を、重複のない当該要素の順位番号と決定し、その一方、これらが異なる場合には、カウントアップされた値番号カウンタの値を、重複のない当該要素の順位番号と決定して、当該順位番号を更新する順位番号更新手段とを備えている。この実施態様によれば、配列の要素に付与された順位番号を、要素の重複を排除した状態のものに変換することが可能となる。

【００１２】また、本発明の目的は、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した配列のソート方法であって、（ａ）メモリモジュールにおいて、自己が把握する配列の部分を構成する要素をソートするステップと、（ｂ）前記自己の把握する前記部分が配列中に占める位置にしたがって、前記配列の部分を把握するメモリモジュールのうち、要素および順位番号を送出する側の提示メモリモジュール、および、要素および順位番号を受理する側の判定メモリモジュールを決定するステップと、（ｃ）提示メモリモジュールにおいて、ソートされた要素を、その順位番号とともに、他のメモリモジュールに所定のバスを介して伝達するステップと、（ｄ）判定メモリモジュールにおいて、所定のバスを介して他のメモリモジュールからの前記要素および順位番号を受理するステップと、（ｅ）前記判定メモリモジュールにおいて、当該判定メモリモジュールが把握する要素の順位番号に基づ

き、受理した要素の順位番号の候補を示す仮想順位番号を算出して、当該仮想順位番号を、前記提示メモリモジュールに返送するステップと、（ｆ）前記提示メモリモジュールにおいて、前記仮想順位番号を受理した場合に、当該仮想順位番号にしたがって、要素の順位番号を更新するステップと、（ｇ）前記ステップ（ｄ）～（ｆ）が終了するごとに、当該ステップ（ｄ）～（ｆ）により所定の順位番号が付された要素に関する提示メモリモジュールと判定メモリモジュールからなるメモリモジュール群を、それぞれ、提示メモリモジュール群、および、判定メモリモジュール群の一方として、ステップ（ｄ）～（ｆ）を繰り返すことにより、各メモリモジュール群における要素の順位番号を更新することにより、配列の各要素の順位番号を確定することを特徴とするソート方法によっても達成される。

【００１３】上記発明によれば、提示メモリモジュールにおける演算、提示メモリモジュールにおける要素および順位番号の送出、判定メモリモジュール群における演算、提示メモリモジュールにおける仮想順位番号の送出が、並列的に実行でき、かつ、バスの衝突も回避することができる。すなわち、これにより、著しく高速に、ソート処理（配列の要素への順位番号付与）を実現することが可能となる。また、使用するメモリ量も、Ｏ（Ｎ）に抑制することが可能となる。上記発明の好ましい実施態様においては、ステップ（ｅ）が、（e1）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号とに基づき、仮想順位番号を算出するステップを含む。また、さらに好ましい実施態様においては、ステップ（ｆ）が、（f1）受理した仮想順位番号を、ステップ（ｃ）にて送出した要素の順位番号に代入するステップを含む。

【００１４】本発明の好ましい実施態様においては、さらに、（ｈ）提示メモリモジュール群において、当該提示メモリモジュール群を構成するメモリモジュールにて把握されている要素が、当該メモリモジュール群においていくつ存在しているかを示す重複数を算出するステップを備え、前記ステップ（ｃ）が、（c1）同一の要素を重複して伝達しないように、ソートされた要素を、その順位番号および重複数とともに、他のメモリモジュールに伝達するステップを含み、前記ステップ（ｅ）が、（e2）受理した要素より前方に位置すべき要素の数を示す前方挿入数と、前方に位置すべき要素に関する順位番号と、受理した順位番号および重複数とに基づき、仮想順位番号を算出するステップを含み、かつ、前記ステップ（ｆ）が、（f2）仮想順位番号と、ステップ（ｃ）における要素の送出時の順位番号との差に基づき、当該要素と同一の要素の順位番号を決定するステップを含む。

【００１５】この実施態様によれば、提示メモリモジュールは、同じ要素を何度も送出する必要がない。また、ある要素の重複数を算出すると、当該要素の順位番号および重複数を、判定メモリモジュールに送信し、判定メモリモジュールでは、当該要素にかかる仮想順位番号の算出を実行することができる。すなわち、これにより、メモリモジュールの利用効率を下げることを防止できる。

【００１６】さらに好ましい実施態様においては、初期的に提示メモリモジュールが単独のメモリモジュールであり、かつ、受信モジュールも単独のメモリモジュールであり、（ｄ）～（ｆ）のステップが終了する毎に、ｎ（ｎ：１以上の整数）がインクリメントされるような2^nのメモリモジュールからなる提示メモリモジュール群と、2^nのメモリモジュールからなる判定メモリモジュール群が形成される。上述したように、2^nのメモリモジュールを用いると、好適に、ソート処理を実現することが可能となる。

【００１７】また、本発明の別の実施態様においては、上記ソート方法により、配列をソートし、かつ、当該ソートされた配列に基づき、前記配列中の要素が、重複なく、かつ、所定の順序にて配置された新たな配列を生成するコンパイル方法は、（ｉ）所定のメモリモジュールにおいて、順位番号にしたがって処理対象となる要素を送出するステップと、（ｊ）前回の処理対象となった要素と同一の要素が送出された場合には、同一の要素の存在数を示す同一値個数カウンタをカウントアップし、その一方、前回の処理対象となった要素と異なる要素が送出された場合には、前回の処理対象となった要素、および、当該要素に関する同一値個数カウンタの値を関連付けて、これらを送出するステップと、（ｋ）前回の処理対象となった要素、および、関連する同一値カウンタの値を受理して、これらを関連付けて新たな配列中に配置するステップとを備え、（ｌ）ステップ（ｉ）～（ｊ）を繰り返すことにより、前記新たな配列中に、要素およびその存在数が関連付けられて配置されることを特徴とする。

【００１８】また、上記コンパイル方法は、さらに、（ｍ）何れかのモジュールにおいて、ステップ（ｊ）にて送出される要素および関連する同一値個数カウンタの値をモニターするステップを備え、当該何れかのモジュールにより、ステップ（ｋ）が実行されても良い。

【００１９】また、上記コンパイル方法は、（ｎ）当該配列の要素を把握するメモリモジュールにおいて、処理対象となっている要素の順位番号および当該要素の存在数をそれぞれ格納する順位番号カウンタおよび同一値個数カウンタを設けるとともに、および、前回の処理対象となった要素を一時的に格納するレジスタを設けるステップと、（ｏ）順位番号にしたがって、当該順位番号が付された要素を把握するメモリモジュールにおいて、当該要素を第１のバスに送出するステップと、（ｐ）配列の要素を把握するメモリモジュールにおいて、受理した

要素とレジスタの内容とを比較して、これらが一致する場合には、存在数をカウントアップする一方、これらが一致しない場合には、レジスタの内容および存在数カウンタの値を、第２のバスに送出した後に、レジスタの内容および存在数カウンタの値を更新するステップと、

（ｑ）何れかのメモリモジュールにおいて、前記レジスタの内容および存在数カウンタの値を、それぞれ要素および当該要素の存在数として、配列中に配置するステップとを備えても良い。

【００２０】ステップ（ｎ）は、さらに、（ｎ1）処理対象となっている要素に関して、重複のない順位番号を格納する値カウンタを設けるステップを含み、前記ステップ（ｐ）は、（ｐ1）受理した要素とレジスタの内容とを比較して、これらが一致する場合に、当該処理対象となる要素の順位番号に、値番号カウンタの値を付与する一方、これらが一致しない場合に、値番号カウンタをカウントアップし、処理対象となる要素の順位番号に、カウントアップされた値番号カウンタの値を付与するステップを含むのが、さらに望ましい。

【００２１】また、本発明の別の実施態様において、上記ソート方法および上記コンパイル方法を用いて、複数の配列の共有化を実現する配列のジョイン方法は、

（ｒ）複数の配列を合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行するステップと、（ｓ）前記合併した配列中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな配列を生成するステップとを備えている。すなわち、所望の配列を併合した状態で、本発明にかかるソート方法およびコンパイル方法を施すことにより、要素の重複を排除した、ジョインされた配列を得ることが可能となる。

【００２２】さらに別の実施態様において、ＣＰＵモジュールと、それぞれがＭＰＵおよびＲＡＭコアとを有する複数のメモリモジュールと、前記ＣＰＵとメモリモジュールとの接続、および／または、任意のメモリモジュール間の接続をなす複数組のバスとを備え、前記一以上のメモリモジュールにより把握される配列に関する処理が、ＣＰＵから各メモリモジュールのＭＰＵに与えられるインストラクションにより、各メモリモジュールのＭＰＵの作動により実行されるように構成された分散メモリ型情報処理システムを利用した複数の配列のジョイン方法は、

【００２３】上記ソート方法および上記コンパイル方法を用いており、かつ、メモリモジュールが、それぞれ、レコード番号に基づき、要素を格納した配列である値リストにおける所定の要素を指定するために、レコード番号に対応する位置に、値リストを示すポインタ値を配置したポインタ配列を備え、（ｒ1）複数の値リストを合併して、これら配列の要素の各々に順位番号を付す前記ソート方法にかかる処理を実行するステップと、（ｔ）前記合併した値リスト中の要素およびその順位番号にしたがって、前記コンパイル方法にかかる処理を実行し、重複した要素の存在しない新たな値リストを生成するとともに、前記要素の順位番号を、重複した要素の存在しない場合の当該要素の順位番号に更新するステップと、

（ｕ）前記重複した要素の存在しない場合の要素の順位番号からなる配列を、新たな値リストを示すための、新たなポインタ配列とするステップとを備えている。

【００２４】

【発明の実施の形態】［ハードウェア構成］以下、添付図面を参照して、本発明の実施の形態につき説明を加える。図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。図１に示すように、コンピュータシステム１０は、単一命令による並列演算を実現するＣＰＵモジュール１２と、並列演算のために必要な種々のデータを記憶するメモリモジュール１４－１、１４－２、１４－３、…と、必要なプログラムやデータを記憶する固定記憶装置１６と、キーボードやマウスなどの入力装置１８と、ＣＲＴなどからなる表示装置２０と、種々の形式のデータ等が記憶されているレガシーメモリ２２とを備えている。また、バス２４－１、２４－２、…において、ＣＰＵモジュール１２、各メモリモジュール１４との接点には、スイッチ２８－１、２８－２、２８－３、…などが配設され、選択された回路要素間における情報の授受が可能となっている。また、ＣＰＵモジュール１２とメモリモジュール１４－１との間、隣接するメモリモジュール間において、バスの連結および接続をなすためのスイッチ３０－１、３０－２、…が設けられている。また、メモリモジュールの入力端子とバスとの接点と、当該メモリモジュールの出力端子とバスとの接点との間に、スイッチ（符号２９参照）が設けられていても良い。図１において、上記スイッチは、破線の丸印にて示されている。

【００２５】さらに、メモリモジュール１４には、単一の入力端子および単一の出力端子だけではなく、一以上の他の端子（入出力端子など）が設けられているのが望ましい。たとえば、後述する第２の実施の形態や第３の実施の形態においては、３つ以上の端子からの入出力を用いて処理が実現されている。

【００２６】ＣＰＵモジュール１２と、メモリモジュール１４との間には、複数のバス２４－１、２４－２、２４－３、２４－４、…とが設けられている。したがって、ＣＰＵモジュール１２とメモリモジュール１４との間、および、メモリモジュール間は、上記バスによりデータ等の授受が可能となっている。また、ＣＰＵ１２と、メモリモジュール１４との間には、制御信号ライン２５が設けられ、ＣＰＵ１２から発せられるインストラクションなどが、全てのメモリモジュール１４に伝達されるようになっている。

【００２７】さらに、ＣＰＵ１２と、他の構成要素（た

とえば、固定記憶装置１６、入力装置１８など）との間には、ローカルバス２６が配設されており、これらの間でもデータ等の授受が可能となっている。ＣＰＵ１２は、固定記憶装置１６に記憶され、或いは、バス２６上に接続されたＲＡＭのような他の記憶装置（図示せず）に記憶されたプログラムを読み出し、このプログラムにしたがって、以下に示すメモリモジュール１４へのインストラクションの送出を含むデータの授受のほか、スイッチ２８～３０の制御等を実行する。また、ＣＰＵ１２は、プログラムにしたがって、レガシーメモリ２２に記憶された種々の形式のデータを受け入れて、この形式のデータを、ＣＰＵ１２、メモリモジュール１４、バス２４からなる系にて処理可能な一連のデータ（配列）に変換し、これらを、各メモリモジュール１４に記憶させることもできる。

【００２８】図２は、各メモリモジュール１４の概略を示すブロックダイヤグラムである。図２に示すように、メモリモジュール１４は、ＣＰＵモジュール１２から与えられるクロックなど同期信号を受け入れるクロックバッファ３２と、データを記憶するＲＡＭコア３４と、後述する空間ＩＤやデータの要素番号等を把握し、ＣＰＵ１２からのインストラクションなどを受理した場合に、空間ＩＤや要素番号に基づき、ＲＡＭコア３４へのデータ書き込みやＲＡＭコアからのデータ読み出しを制御するＭＰＵ３６と、バスの何れかからのデータを受け入れて、ＲＡＭコア３４に供給し、および／または、ＲＡＭコア３４からのデータを何れかのバスに送出するＩ／Ｏ３８とを有している。この実施の形態において、メモリモジュール１４は、制御信号ライン２５を介して、ＣＰＵからのインストラクションを受け入れ、ＭＰＵ３６が、このインストラクションに応答して、ＲＡＭコア３４のデータを読み出し、ＲＡＭコア３４にデータを書き込み、或いは、データに所定の処理を施すことができるようになっている。また、ＲＡＭコア３４へのデータアクセスや、Ｉ／Ｏを介してデータ入力およびデータ出力は、クロックバッファ３２に与えられるクロックなどの同期信号に基づき実行される。上記メモリモジュール１４のＭＰＵ３６は、複数の処理ユニットからなり、並列的に複数の処理を実行できるのが望ましい。

【００２９】図１および図２から明らかなように、本発明において、コンピュータシステム１０は、メモリ共有型のシステムであると考えることができる。また、後述するように、制御信号ライン２５を介して、各メモリモジュール１４にインストラクションを与えることにより、各メモリモジュール１４が並列的に処理を実行する。また、バスへのデータ出力およびバスからのデータ入力などが、所定の同期信号に基づき実行される。したがって、このコンピュータシステム１０は、ＳＩＭＤの形態をなしていると考えることができる。このような構成を備えたコンピュータシステム１０は、基本的には、本発明者の考案にかかる、特願平１１－２６３７９３号に記載された多空間メモリ、メモリモジュールおよび組み替え可能バスを備えている。これらにつき、以下に簡単に説明を加える。

【００３０】（１）多空間メモリ

本明細書において、多空間メモリとは、メモリ空間を、空間ＩＤとアドレスとに基づきアクセスするために割り当てられたメモリ空間をいう。これにより、一連のデータが多数のプロセッサに分掌されていても、各プロセッサが、これを確実に分離、認識することができる。

【００３１】従来のメモリ空間においては、プロセス毎に個別の領域を割り当てることはあっても、一連の変数（配列、構造体など）毎にメモリ空間を割り当てることは行われてこなかった。したがって、以下、このような従来のメモリ空間を「単一メモリ空間」と称する。単一メモリ空間のシステムにおいては、アドレスのみを用いてデータにアクセスしているため、関連を有する一連のデータを分離したり、認識することができなかった。このため、実際には並列処理が可能であっても、その可否を判断できない場合が多かった。また、ある単一メモリ空間に、新たな一連のデータを収容させる場合に、当該一連のデータの収容場所を確保するために、ガーベージコレクションを実行する必要があった。

【００３２】これに対して、本発明においては、メモリ空間に、空間ＩＤを導入し、一連のデータについて同一のＩＤを付与している。また、メモリモジュール１４において、自身のＲＡＭコア３４に保持されているデータに関する空間ＩＤを把握し、これにより、各メモリモジュール１４自体が、現在アクセスされているデータの空間ＩＤを参照することにより、自己の作動の是非を決定することができる。また、各メモリモジュールが空間ＩＤと関連付けて、一連のデータの全部或いは一部を保持できるため、ある一連のデータを、複数のメモリモジュール１４に分割して記憶させることができ、これによりガーベージコレクションを不要にすることができる。

【００３３】（２）メモリモジュール

また、本発明においては、各メモリモジュール１４が、ＭＰＵ３６を有し、上記空間ＩＤのほか、自己が保持する一連のデータの各々の要素番号を把握している。したがって、ＣＰＵ１２からのインストラクションを受理した後、ＭＰＵ３６が、インストラクションにしたがってアクセスすべきデータが、自己のＲＡＭコア３４中に保持されているものか否かを判断して、アクセスに必要の是非を決定することができる。さらに、各メモリモジュール１４が、自己のＲＡＭコア３４に格納されている配列要素の添え字の範囲から、ＳＩＭＤでのインストラクションにおける暗黙の処理の分担範囲を決定することが可能である。各メモリモジュール１４は、ＣＰＵ１２からのインストラクションにしたがって、処理すべき要素の記憶順序を入れ替えて、自己のＲＡＭコア３４中に保

持している要素をソートすることが可能である。

【００３４】（３）組み替え可能バス

本発明においては、ＣＰＵ１２が、スイッチ２８－１、２８－２、…およびスイッチ３０－１、３０－２、…を選択的にオン／オフして、データの授受をなすべきメモリモジュール１４を指定することにより、パイプライン処理を実現している。たとえば、図３に示すように、あるメモリモジュール１４－ｉから出力されたデータを、他のメモリモジュール１４－ｊに与え、かつ、当該他のメモリモジュール１４－ｊから出力されたデータを、さらに他のメモリモジュール１４－ｋに伝達すべき場合には、ＣＰＵ１２は、バス２４－ｍを、メモリモジュール１４－ｉ、１４－ｊのために割り当て、かつ、バス２４－ｎを、メモリモジュール１４－ｊ、１４－ｋのために割り当てるように、各スイッチの状態を設定する。

【００３５】さらに、これらパイプライン処理は、単一のメモリモジュール間の接続により実現される場合だけでなく、複数の一連のメモリモジュール（メモリモジュール群）の間の接続により実現することも可能である。達成しようとする処理に応じて、各メモリモジュール間をつなぎ替え、各接続経路毎に、定められた種類のデータを定められた順序にて一方向に連続転送することで、バスの能力を１００％近く使用できるように、通信をスケジュール化することができる。これにより、分散メモリ型の並列処理システムの最大の問題であった、プロセッサ間通信のパフォーマンスの低さを、解消することができる。

【００３６】［多空間メモリ］再度、多空間メモリを用いた、本発明にかかるコンピュータシステムにおける各メモリモジュールのメモリ管理、および、インストラクションにしたがったメモリアクセスにつき、より詳細に説明を加える。図４は、多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。図４（ａ）に示すように、メモリモジュール１４中のＲＡＭコア３４には、空間ＩＤ管理テーブルが設けられる。これにより、メモリモジュール１４のＭＰＵ３６は、自己が保持するデータの空間ＩＤ等必要な情報を把握することが可能となる。

【００３７】図４（ｂ）に示すように、空間ＩＤ管理テーブルには、自己が保持するデータ群ごとの、空間ＩＤ、ＣＰＵの管理の下での、データ群の論理開始アドレス、データ群が割り付けられた領域のサイズ、ＲＡＭコア３４中の物理開始アドレス、当該空間ＩＤを有する一連のデータの全サイズ、および、アクセス制限を示すアクセス制限フラグが格納されている。アクセス制限フラグは、この実施の形態においては、読み出しのみ可能（Ｒ）、書き込みのみ可能（Ｒ）、読み書き可能（ＲＷ）の３つの状態を示すことができるようになっている。メモリモジュール１４のＭＰＵ３６は、ある空間ＩＤを有するデータ群が与えられた際に、ＲＡＭコア３４中に当該データ群を収容すべき、１以上の領域を見出して、当該領域にデータ群をそのまま、或いは、２以上に分割して収容する。この際に、与えられた空間ＩＤ、論理開始アドレス、全サイズ、アクセス制限フラグとともに、実際にデータを収容したＲＡＭコア中の論理開始アドレスや、割り付け領域サイズも、空間ＩＤ管理テーブルに記憶される。図４（ｃ）は、図４（ｂ）による空間ＩＤ管理テーブルにしたがったＲＡＭコア３６中のデータを示す図である。

【００３８】［メモリアクセスの概略］このように構成されたメモリモジュール１４へのアクセスにつき以下に説明を加える。図５に示すように、まず、ＣＰＵ１２が、空間ＩＤおよび論理アドレス、並びに、必要なインストラクション（たとえば、データの書き込みや読み出し）を、制御信号ライン２５を介して、全てのメモリモジュール１４に伝達する。各メモリモジュール１４においては、これに応答して、ＭＰＵ３６に設けられた空間コンパレータ５２が、空間ＩＤと、自己の空間ＩＤ管理テーブル上に保持されている空間ＩＤとを比較して、同一のものを、自己が保持しているかを判断し、また、、アドレスコンパレータ５４が、論理アドレスについて、同様の判断を行う。次いで、メモリモジュール１４のＭＰＵ３６が、自己のＲＡＭコア３４に、インストラクションによる処理対象となるデータが保持されていると判断した場合には、アドレスカリキュレータ５６が、空間ＩＤ管理テーブルを参照して、ＲＡＭコア３４中の物理アドレスを算出し、処理対象となるデータを特定する。このようにして、データが特定された後に、ＭＰＵ３６は、ＣＰＵ１２から与えられたインストラクションに応じた処理（たとえば、データの書き込みや読み出し）を実行し、必要な場合には、データをＣＰＵ１２に伝達する（図５（ｃ）参照）。

【００３９】［ソート処理（第１の実施の形態）］このように構成されたコンピュータシステム１０にかかるソート処理につき説明を加える。なお、以下の説明において、本発明にかかるメモリモジュールが、ＭＰＵ（プロセッサ）を備えたメモリモジュールであることから、ＰＭＭ（Processor Memory Module）と称する。

【００４０】理解を容易にするために、図６に示すように、４つのＰＭＭが、それぞれ、２つの要素（名字）を保持している場合を考える。図６（ａ）に示すように、あるＰＭＭ（第１のＰＭＭ１４－１）には、要素の添え字（すなわちレコード番号）が「０」である「やまもと」という姓と、添え字が「１」である「あべ」という姓とが保持されている。第２のＰＭＭ１４－２には、添え字が「２」である「いとう」という姓と、添え字が「３」である「すぎもと」という姓とが保持されている。以下、第３のＰＭＭ１４－３、第４のＰＭＭ１４－４にも、それぞれ、図６（ａ）に示すような添え字に対応した姓が保持されている。これら要素からなる配列に

は、同一の空間ＩＤが付され、各ＰＭＭのＭＰＵ３６は、その空間ＩＤ管理テーブルを利用して、自己のＲＡＭコア３４が管理する要素の添え字（レコード番号）や実際に格納している物理アドレス等を管理している。

【００４１】たとえば、ＣＰＵ１２から、制御信号ライン２５を介して、この空間ＩＤを有する配列のソートをするインストラクションが、各ＰＭＭ１４－１～１４－４に与えられたと考える。図７は、本実施の形態にかかるソート処理の処理手順を示すフローチャートである。図７に示すように、インストラクション（たとえば、「ある空間ＩＤを有する配列中の要素をソートせよ」というインストラクション）がＣＰＵ１２により発行されると（ステップ７００）、このインストラクションに応答して、各ＰＭＭにおいて、各ＰＭＭのＭＰＵ３６は、制御信号ライン２５を介して与えられたインストラクションを受理して、その内容を解釈し（ステップ７０１）、インストラクション中の「空間ＩＤ」を調べ（ステップ７０２）、自己のＲＡＭコア３４が保持するデータの空間ＩＤに関連しているか否かを判断する（ステップ７０３）。ステップ７０３にてノー（No）と判断された場合には、処理を終了し、その一方、イエス（Yes）と判断された場合には、ＭＰＵ３６は、空間ＩＤ管理テーブルを参照して、当該空間ＩＤに関するデータ群が書き込み可能な状態になっているかなど、必要なチェックを行う（ステップ７０４）。チェックによって異常があると判断された場合（ステップ７０５でイエス(Yes)）には、ＭＰＵ３６は、制御信号ライン２５を介してエラーが生じたことをＣＰＵ１２に通知する。その一方、異常がない場合には、ＭＰＵ３６は、以下に述べるソート処理本体を実行する（ステップ７０７以下）。

【００４２】まず、処理に関連するＰＭＭ１４－１～１４－４の各々は、自己の保持する要素のソートを実行する（ステップ７０７）。このソートは、実際に、各ＰＭＭ１４中の要素の入れ替えを伴う。より具体的には、ＭＰＵ３６が、自己のＲＡＭコア３４中に保持された要素を、クイックソートなど既知のソート手法を用いてソートする。図６（ｂ）は、図６（ａ）に示す各ＰＭＭ中の配列中の要素がソートされた状態を示す図である。なお、図６（ｂ）に示すように、上記要素のソートにともなって、各要素の添え字（レコード番号）の配置も変更されていることに留意すべきである。

【００４３】次いで、各ＰＭＭ１４のＭＰＵ３６は、自己が保持／管理している配列中の要素の数だけ、順位番号を配置するための領域（順位番号領域）を確保し、各順位番号の初期値を与える（ステップ７０８）。図６（ｃ）は、各ＰＭＭに関して、順位番号の初期値が与えられた状態を示す図である。このように、初期的に、順位番号は、各モジュール内にてソートされた要素内で付与される。次いで、隣接するペア間のマージおよび順位番号付与が実行される（ステップ７０９）。ステップ７０９において、まずＣＰＵ１２が、バス２４上のスイッチ２８、３０を制御して、ソート処理に関連するＰＭＭのうち、所定のペアの一方の入力と他方の出力、並びに、一方の出力と他方の入力とを接続する。上記ペアは隣接している２つのＰＭＭ、隣接していない場合にも近傍に位置する２つのＰＭＭからなるのが望ましい。たとえば、図１において、ソート処理に関連するものが、ＰＭＭ１４－１～１４－４である場合には、ＰＭＭ１４－１および１４－２をペアとして、かつ、ＰＭＭ１４－３および１４－４をペアとするのが望ましい。ＣＰＵ１２は、たとえば、図８に示すように、バス２４－１にＰＭＭ１４－１の出力およびＰＭＭ１４－２を接続するように、かつ、バス２４－２にＰＭＭ１４－１の入力とＰＭＭ１４－２の出力とを接続するように、スイッチ２８を制御し、かつ、バス２４－１にＰＭＭ１４－３の出力およびＰＭＭ１４－４を接続するように、かつ、バス２４－２にＰＭＭ１４－３の入力とＰＭＭ１４－４の出力とを接続するように、スイッチ２８を制御する。さらに、ＣＰＵ１２は、さらに、ＰＭＭ１４－２とＰＭＭ１４－３との間に配置されるバス２４－１、２４－２上のスイッチ３０－５、３０－６をオフにする。図８において、黒丸で表わしているものが導通している状態を示し、白丸で表わしているものが導通ないしＰＭＭと接続されていない状態を示している。また、他のものは他のＰＭＭ（図示せず）の状態に従っている。なお、図８の例では、バス２４－１、２４－２を、スイッチ３０－５、３０－６をオフにすることにより分割して、バスをより有効に利用していることが理解できるであろう。

【００４４】このようにして、図９に模式的に示すように、ＣＰＵ１２によってＰＭＭ間の接続が規定されると、ＰＭＭのペア間での順位番号付与の処理本体が実行される。図１０ないし図１２は、理解を容易にするために図６にて示した配列に関する順位番号付与を模式的に示す図であり、図１３は、より一般的な、ＰＭＭのペア間の順位番号処理を示すフローチャートである。　図１０ないし図１２においては、ＰＭＭ１４－１およびＰＭＭ１４－２における処理過程のみを示したが、ＰＭＭ１４－３およびＰＭＭ１４－４における処理も、並列的に実行されている。なお、ここで、処理において、最初にデータを他方のＰＭＭに与えるものを前半のＰＭＭと称し、受理するもの（他方のＰＭＭ）を後半のＰＭＭと称する。前半のＰＭＭは、要素や順位番号を提示するため提示ＰＭＭということができ、その一方、後半のＰＭＭは、提示された順位番号を判定するため判定ＰＭＭということができる。ペアのうち何れのＰＭＭが前半のＰＭＭとなっても良い。この例では、便宜的に、ＰＭＭ１４－１が前半のＰＭＭとなり、ＰＭＭ１４－２が後半のＰＭＭとなっている。

【００４５】まず、前半のＰＭＭにおいては、処理位置を示すポインタ（以下、「ＰＵＴポインタ」と称する）

を初期位置（ソートされた配列の部分において先頭つまり「０」番目の位置）に配置する。その一方、後半のＰＭＭにおいて、以下に説明するように、前半のＰＭＭから受理する要素と、まず比較すべき位置などを示すポインタ（以下、「比較ポインタ」と称する）を初期位置（ソートされた配列の部分において先頭つまり「０」番目の位置）に配置する（図１０（ａ）、および、図１３のステップ１３０１、１３１１参照）。本実施の形態において、後半のＰＭＭにて使用する比較ポインタは、（Ｘ、Ｙ、Ｚ）という構造体配列の形態をとっている。ここに、Ｘは、比較すべき先頭位置（つまり、未比較の要素の先頭位置、以後「未処理位置」と称する）を示し、Ｙは、前半のＰＭＭから受信した要素の総数を示し（以下、場合によって「前方挿入数」と称する。）、Ｚは、前半のＰＭＭと後半のＰＭＭとをマージして得られた仮想的な配列における、前半のＰＭＭから与えられた要素の順位番号の案（以下、場合によって「仮想順位番号」と称する。）を示す。

【００４６】次いで、前半のＰＭＭのＭＰＵにより、最初のデータ転送が実行される。このデータ転送では、ＰＵＴポインタが示す位置の要素が、バスを介して、後半のＰＭＭに伝達される（図１０（ｂ）およびステップ１３０３、１３１２参照）。なお、ステップ１３０２の分岐では、２つのＰＭＭ間の処理では常にイエス（Yes）と判断されるが、これについては後述する。最初のデータ転送では、要素「あべ」が後半のＰＭＭに伝達される。後半のＰＭＭにおいては、後半のＰＭＭに格納されている配列の部分において、伝達された要素「あべ」を挿入すべき位置を捜し出す（ステップ１３１３）。これは実際に値を挿入するのではなく、挿入すべき位置を捜し出せば良い。本実施の形態において、各ＰＭＭの配列の部分に格納された要素は、実際にソートされた状態で配置されている。したがって、挿入位置の検索は、バイセクション法（二分割法）など、高速検索手法を用いて実現することができる。挿入位置を捜し出すことにより、順位が確定していない要素であって、挿入位置より前方に位置する要素の範囲（以下、「範囲１」と称する）を特定することが可能となる。なお、本実施の形態において、同じ要素があった場合には、前半のＰＭＭの順位が優先するという取り決めをしている。したがって、前半のＰＭＭから伝達された要素「あべ」が、後半のＰＭＭにも存在する場合には、前半ＰＭＭに格納されている方の順位が優先（つまり、より小さな順位番号）となる。

【００４７】本例では、前半のＰＭＭから伝達された要素「あべ」は、後半のＰＭＭが把握する配列の部分中、要素「あべ」の前方に位置することがわかり、これにより、範囲「１」に属する要素が存在しないことがわかる（図１０（ｃ）参照、および、ステップ１３１４においてイエス(Yes)）。そこで、後半のＰＭＭのＭＰＵは、伝達された要素「あべ」の順位番号として「０（すなわち先頭）」を、他方のバスを介して、前半のＰＭＭに返送する（ステップ１３１５）。次いで、後半のＰＭＭのＭＰＵは、前方挿入数をインクリメントして「１」にするとともに、仮想順位番号をインクリメントして「１」にする（ステップ１３１６）。これは、前方のＰＭＭから伝達された要素が一つ増加したため、前方挿入数をインクリメントする必要があり、かつ、次の要素の順位番号は、少なくとも、今回与えたもの（この場合では「０」）をインクリメントする必要があるからである。後半のＰＭＭから要素の順位番号（挿入位置）が与えられると（ステップ１３３２）、前半のＰＭＭのＭＰＵは、与えられた順位番号を、該当する要素の順位番号として格納し（ステップ１３３４）、次いで、ＰＵＴポインタをインクリメントする（図１１（ａ）、および、ステップ１３３５参照）。このようにして、前半のＰＭＭ中のある要素の順位が確定する。

【００４８】次に、前半のＰＭＭのＭＰＵは、ＰＵＴポインタが示す位置の要素「やまもと」を、バスを介して、後半のＰＭＭに伝達する（図１１（ｂ）およびステップ１３０３参照）。後半のＰＭＭにおいては、先に、要素「あべ」が伝達されたときと同様に、伝達された要素「やまもと」を挿入すべき位置を捜し出す（ステップ１３１３）。要素「やまもと」は、後半のＰＭＭが把握する配列の部分中、要素「はら」の後方に位置することがわかる（図１１（ｃ）、および、ステップ１３１４にてイエス(Yes)）。これにより、後半のＰＭＭが把握する配列の部分において、要素「はら」およびその前方に位置する要素の数、並びに、各要素の順位を確定することができる。より詳細には、後半のＰＭＭのＭＰＵは以下の手順にて、上記要素の順位を確定させる。

【００４９】まず、範囲「１」に含まれる要素に関する順位番号に、それぞれ、前方挿入数「Ｙ」が加えられる（ステップ１３１７）。これにより、範囲「１」に含まれる要素の順位が確定する。前述した例では、要素「あべ」の順位番号は「０＋１＝１」、要素「はら」の順位番号は「１＋１＝２」となる。次いで、範囲１に含まれる要素のうち、最後尾の要素の順位番号が、仮想番号に代入されるとともに（ステップ１３１８）、未処理位置を、範囲１における最後尾の要素の次の要素の位置に変更する（ステップ１３１９）。上記例では、要素「はら」の順位番号「２」が、比較ポインタ（構造体配列）のＺに与えられ、かつ、未処理位置が、「０」から「２」に変更される。これにより、構造体配列は（２、１、２）となる。このような処理の後、構造体配列中の、前方挿入数「Ｙ」および仮想順位番号「Ｚ」がインクリメントされる（ステップ１３２０）。これにより、構造体配列は、（２、２、３）となる（図１１（ｄ）参照）。ステップ１３２０にて得られた仮想順位番号が、ステップ１３１２にて受信した要素（上記例では、「や

まもと」）の順位番号となり、後半のＰＭＭのＭＰＵは、当該順位番号（上記例では「３」）を前半のＰＭＭに伝達する（ステップ１３２１）。このような処理の後に、さらに、仮想順位番号がインクリメントされる（ステップ１３２２）。これは、次の要素の順位番号は、少なくとも、今回与えた順位番号よりも１つ大きなものとなるからである。

【００５０】前半のＰＭＭは、受理した順位番号を、該当する要素の順位番号として格納し、次いで、ＰＵＴポインタをインクリメントする。このようにして、前半のＰＭＭにおける要素の順位番号が確定する。前半のＰＭＭにおいて、未処理の要素が既に存在しない（すなわち、すべての要素について順位番号が確定し、ＰＵＴポインタの位置には要素が配置されていない）には、前半のＰＭＭのＭＰＵは、終了を示す値を、後半のＰＭＭに伝達する（ステップ１３０６参照）。ここに、終了を示す値は、配列の最後尾の要素を示す値よりも大きな値である。後半のＰＭＭは、上記終了を示す値の受理に応答して、略同様の処理（図１３のステップ１３１２～１３２２）の処理を実行する。上記例では、終了を示す値の受理にもかかわらず、範囲「１」に含まれる要素が存在しないため、ステップ１３１５、１３１６を介してステップ１３２３に達し、処理を終了する（図１２（ｂ）参照）。

【００５１】前半のＰＭＭにおいては、終了を示す値の送出（ステップ１３１６参照）により、および、全ての要素の順位番号の確定（ステップ１３３６でイエス(Yes)）により処理が終了する。上記処理と同様の手順にて、ＰＭＭ１４－３とＰＭＭ１４－４との間でも、マージ処理が実行され、これにより、図１２（ｃ）に示すように各要素の順位番号が確定する。

【００５２】２つのＰＭＭにおける各要素の順序番号が確定すると、ＣＰＵ１２は、スイッチを切り換えて、各々が２つのＰＭＭからなる、２つのＰＭＭ群の間を接続する。図１４（ａ）および図１４（ｂ）は、それぞれ、図８に示すＰＭＭにおける、２つのＰＭＭ群の接続の一例を示す図である。図１４（ａ）においては、ＰＭＭ１４－１、１４－２が、第１のＰＭＭ群を構成し、ＣＰＵ１２は、ＰＭＭ１４－３、１４－４が第２のＰＭＭ群を構成し、ＰＭＭ１４－１、１４－２の出力と、ＰＭＭ群１４－３の入力とが接続され、ＰＭＭ１４－３の出力とＰＭＭ１４－４の入力とが接続され、かつ、ＰＭＭ１４－４の出力とＰＭＭ１４－１、１４－２の入力とが接続されるように、スイッチ２８、３０を制御する（図７のステップ７０９参照）。或いは、図１４（ｂ）に示すように、ＰＭＭ１４－１、１４－２の出力が、ＰＭＭ１４－３、１４－４と接続されるようにスイッチが制御されても良い。

【００５３】図１５（ａ）、（ｂ）は、それぞれ、図１４（ａ）、（ｂ）を模式的に表わした図である。後に明らかになるが、図１５（ａ）において、ＰＭＭ１４－４からＰＭＭ１４－１、１４－２に与えられるデータ（図中、符号❶参照）は順位番号を示し、ＰＭＭ１４－１、１４－２からＰＭＭ１４－３に与えられるデータ（図中、符号❷参照）は要素を示し、かつ、ＰＭＭ１４－３からＰＭＭ１４－４に与えられるデータ（図中、符号❸参照）は、要素およびＰＭＭ１４－３が算出した仮想順位番号を示す。また、図１５（ｂ）においても、各ＰＭＭ間において授受されるデータ❶、❷は、図１５（ａ）のものと同じであり、その一方、ＰＭＭ１４－３からＰＭＭ１４－４に伝達されるデータ（符号❸参照）は、ＰＭＭ１４－３が算出した仮想順位番号を示す。

【００５４】上記図１２に示すように、２つのＰＭＭのペアにおける配列の部分およびこれらに含まれる要素の順位番号から、２つのＰＭＭ群におけるマージ処理および配列の順位番号を決定する処理（図７のステップ７０９参照）につき、説明を加える。なお、以下の説明では、図１４（ｂ）、図１５（ｂ）に示すバスの接続態様にしたがって、各ＰＭＭにて実行する処理を説明する。

【００５５】まず、各々がＰＭＭ１４－１およびＰＭＭ１４－２（以下、「前半のＰＭＭ群」と称する。）において、ＰＵＴポインタを初期位置に配置する（ステップ１３０１）。なお、以後の処理では、ＰＵＴポインタは、前半のＰＰＭ群を構成するＰＭＭにおいて、自己の掌握する要素が送出されるのにしたがって移動する。その一方、後半のＰＭＭの各々は、比較ポインタを、その構造体配列を初期化するとともに、初期位置に配置する（ステップ１３０２）。次いで、前半のＰＭＭ群を構成する各ＰＰＭにおいては、現在、前半のＰＭＭ群を構成する各ＰＭＭは、どの順位番号の要素が送出されたかを把握している。なお、フローチャートでは、ＰＵＴポインタとして、送信時に利用する送信ポインタと受信ポインタと双方を用いているが、基本的にこれら送信ポインタおよび受信ポインタの移動は、後半のＰＭＭ群における処理時間を挟むが、僅かな時間差のみをもって行われている。たとえば、後述するように、あるＰＭＭにおいて送信ポインタをインクリメントとした（ステップ１３０４参照）場合には、当該ＰＭＭは、受信処理においても、受信ポインタをインクリメントする（ステップ１３３５参照）。

【００５６】前半のＰＰＭ群を構成する各ＰＭＭは、処理の対象となる要素の順位番号に基づき、当該要素が、自己が掌握しているものか否かを判断する（ステップ１３０２）。このステップ１３０２にてイエス(Yes)と判断された場合には、ＰＵＴポインタの指し示す要素を、バス２４を介して、ＰＭＭ１４－３、１４－４に伝達する（図１３のステップ１３０２、および、図１６（ａ）参照）。上記例では、まず、順位番号「０」である要素「あべ」が、ＰＭＭ１４－１から、ＰＭＭ１４－３およびＰＭＭ１４－４に伝達される。この処理により、ＰＭ

Ｍ１４－１においては、ＰＵＴポインタの位置が移動する（ステップ１３０４）。

【００５７】ＰＭＭ１４－３およびＰＭＭ１４－４は、それぞれ、要素を受信し（ステップ１３１２）、その要素を挿入すべき位置を捜し出し（ステップ１３１３）、範囲「１」に属する要素が存在するか否かを判断する（ステップ１３１４）。上記要素「あべ」に関しては、ステップ１３１４においてノー(No)と判断される。これにより、ＰＭＭ１４－３において、要素「あべ」の仮想順位番号は「０」となるため、この値を、ＰＭＭ１４－４に伝達する。ＰＭＭ１４－４においても、要素「あべ」の仮想順位番号は「０」となる。そこで、ＰＭＭ１４－４のＭＰＵは、「MAX（０、０）＝０」を、要素「あべ」の順位番号として、前方のＰＭＭ群にバスを介して返送する（図１６（ｂ）およびステップ１３１５参照）。次いで、ＰＭＭ１４－３、１４－４においては、構造体配列中の前方挿入数（Ｙ）および仮想順位番号（Ｚ）が、それぞれインクリメントする（ステップ１３１６）。上記例では、これにより、それぞれの構造体配列が、（０、１、１）、（０、１、１）となる。

【００５８】後半のＰＭＭ群から順位番号が与えられると（ステップ１３３１）、前半のＰＭＭ群を構成する各ＰＭＭは、現在処理中の要素（たとえば、要素「あべ」）が、自己が掌握するものであるかを判断する（ステップ１３３３）。要素「あべ」の順位番号が伝達された場合には、ＰＭＭ１４－１が、上記ステップ１３３３でイエス(Yes)と判断し、その位置の要素に対応する順位番号を、後半のＰＭＭ群から与えられたものに書き換える（図１６（ａ）およびステップ１３３４参照）。同様に、前半のＰＭＭ群は、次の順位番号を付与された要素を、後半のＰＭＭ群に伝達する。上記例では、ＰＭＭ１４－２から、要素「あべ」が伝達され（図１７（ａ）参照）、後半のＰＭＭ群の各々の仮想順位番号のうち大きなもの「MAX（１、１）＝１」が、当該要素「あべ」の順位番号として、前半のＰＭＭ群に伝達される（図１７（ｂ）参照）。また、後半のＰＭＭ群を構成するＰＭＭ１４－３、１４－４において、構造体配列は、それぞれ、（０、２、２）、（０、２、２）となる（図１７（ｂ）参照）。

【００５９】さらに、前半のＰＭＭ群は、次の順位番号を付与された要素を、後半のＰＭＭに伝達する。上記例では、ＰＭＭ１４－２から、要素「はら」が伝達される（図１８（ａ）参照）。ＰＭＭ１４－３において、要素「はら」は、当該ＰＭＭ１４－３が掌握する要素「たなか」よりも後ろであると判断される（ステップ１３１３参照）。したがって、ＰＭＭ１４－３においては、範囲「１」には、要素「さとう」および要素「たなか」が属するため、要素「さとう」および要素「たなか」の順位番号に、それぞれ、前方挿入数「Ｙ（＝２）」が加えられる。これにより、要素「さとう」の順位番号は「０＋２＝２」、要素「たなか」の順位番号は「２＋２＝４」と決定される（ステップ１３１７参照）。次いで、ＰＭＭ１４－３のＭＰＵは、構造体配列（現在の値は（０、２、２））の仮想順位番号Ｚに、範囲「１」中の末尾の要素の順位番号「４」を与え（ステップ１３１８参照）、未処理位置を進める（すなわち、Ｘの値を「０」から「２」にする）（ステップ１３１９参照）。さらに、ＰＭＭ１４－３のＭＰＵは、構造体配列（現在の値は（２、２、４）の前方挿入数「Ｙ」および仮想順位番号「Ｚ」をインクリメントする（ステップ１３２１参照）。これにより、構造体配列は、（２、３、５）となる。ＰＭＭ１４－３における仮想順位番号「Ｚ（＝５）」は、バスを介してＰＭＭ１４－４に伝達される。その後に、ＰＭＭ１４－３のＭＰＵは、構造体配列の仮想順位番号「Ｚ」をインクリメントする（ステップ１３２２参照）。上記例では、ステップ１３２２を施すことにより、構造体配列は、（２、３、６）となる。

【００６０】その一方、ＰＭＭ１４－４において、要素「はら」は、当該ＰＭＭ１４－４が掌握する要素「すぎもと」と「よしだ」との間に位置すると判断される（ステップ１３１３参照）。したがって、ＰＭＭ１４－４において、範囲「１」には、要素「すぎもと」が属するため、要素「すぎもと」の順位番号に、前方挿入数「Ｙ（＝２）」が加えられ、これにより、要素「すぎもと」の順位番号は、「１＋２＝３」と決定される（ステップ１３１７参照）。次いで、ＰＭＭ１４－４のＭＰＵは、構造体配列（現在の値は（０、２、２））の仮想順位番号「Ｚ」に、範囲「１」中の末尾の要素の順位番号「３」を与え（ステップ１３１８参照）、未処理位置を進める（すなわち、「Ｘ」の値を「０」から「１」にする）（ステップ１３１９参照）。さらに、ＰＭＭ１４－４のＭＰＵは、構造体配列（現在の値は（１、２、３））の前方挿入数「Ｙ」および仮想順位番号「Ｚ」をインクリメントする（ステップ１３２１参照）。これにより、構造体配列は（１、３、４）となる。

【００６１】この後に、ＰＭＭ１４－４は、ＰＭＭ１４－３から与えられた仮想順位番号「Ｚ（＝５）」と、自己が算出した仮想順位番号「Ｚ（＝４）」とを比較し、より大きな方の値である「MAX（５、４）＝５」を、伝達された要素「はら」の順位として、前半のＰＭＭ群に伝達する（ステップ３２１参照）。これにより、前半のＰＭＭ群において（より詳細には、要素「はら」を送出したＰＭＭ１２－２において）、当該要素の順位番号が「５」であることが確定する。なお、ＰＭＭ１４－４においても、ステップ１３２１の後に、構造体配列中の仮想順位番号「Ｚ」がインクリメントされる（ステップ１３２２参照）。上記例では、構造体配列は、（１、３、５）となる。

【００６２】同様に、前半のＰＭＭ群より要素「やまもと」が送出される（図１９（ａ））が、この場合の処理

も、図１３にしたがって実行される。再度、簡単に説明すると、要素「やまもと」を受理したＰＭＭ１４－３においては、要素「やまもと」の挿入位置より前方に、範囲「１」に属する要素が存在しないため、ＰＭＭ１４－３は、その構造体配列中の仮想順位番号「Ｚ（＝６）」を、ＰＭＭ－４に伝達する。ＰＭＭ１４－４においても、要素「やまもと」の挿入位置より前方に、範囲「１」に属する要素が存在しないため、その構造体配列中の仮想順位番号「Ｚ（＝５）」と、伝達された仮想順位番号「Ｚ（＝６）」とを比較して、その大きな方（ＭＡＸ（６、５）＝６）を、要素「やまもと」の順位番号として、前半のＰＭＭ群に返送する（図１９（ｂ）およびステップ１３１５参照）。前半のＰＭＭ群においては、要素「やまもと」を送出したＰＭＭ１４－１が、要素「やまもと」に対応する順位番号を、受理した順位番号（＝６）に書き換える。なお、ＰＭＭ１４－３においては、ステップ１３１６を経ることにより、その構造体配列は（２、４、７）となり、その一方、ＰＭＭ１４－４においては、ステップ１３１６を経ることにより、その構造体配列は（１、４、６）となる。

【００６３】このようにして、前半のＰＭＭ群において全ての要素の送出が終了すると、前半のＰＭＭ群を構成する、何れかのＰＭＭが、終了を示す値を後半のＰＭＭ群に伝達する（ステップ１３０６参照）。後半のＰＭＭ群を構成する各ＰＭＭは、これを受理して、それぞれ、ステップ１３１２ないしステップ１３２３の処理を実行する。上記例では、ＰＭＭ１３－４においては、順位の確定していない要素「よしだ」が存在する。このため、ＰＭＭ１３－４においては、ステップ１３１４においてイエス(Yes)と判断され、範囲「１」に属する要素「よしだ」の順位番号に、前方挿入数「Ｙ」を加えて「３＋４＝７」、得られた数「７」を、要素「よしだ」の順位番号とする。このような処理を経た後、後半のＰＭＭ群を構成する各ＰＭＭにおいて、ステップ１３２３にてイエス(Yes)と判断され、後半のＰＭＭ群における処理も終了する。

【００６４】上記例では、４つのＰＭＭに配列中の要素が格納されていたが、それ以上のＰＭＭ中に配列中の要素が格納されている場合には、さらに、４つのＰＭＭを一群のＰＭＭとして、各々が４つのＰＭＭからなるＰＭＭ群のペアを作成し、これらペアの間で、略同様の処理を実行すれば良い。たとえば、図２０に示すように、１０２４個のＰＭＭにて、ある配列中の要素が格納されていると考える。この場合には、まず、ＰＭＭ１およびＰＭＭ２、ＰＭＭ３およびＰＭＭ４、ＰＭＭ５およびＰＭＭ６、…ＰＭＭ１０２３およびＰＭＭ１０２４を、それぞれ連結し（ＰＭＭ間の実線参照）、これら２つのＰＭＭ間で、要素の順位番号を確定し、次いで、ＰＭＭ１およびＰＭＭ２を前半のＰＭＭ群、ＰＭＭ３およびＰＭＭ４を後半のＰＭＭ群とするＰＭＭ群のペア、ＰＭＭ５およびＰＭＭ６を前半のＰＭＭ群、ＰＭＭ７およびＰＭＭ８（図示せず）を後半のＰＭＭ群とするＰＭＭ群のペア、…ＰＭＭ１０２１およびＰＭＭ１０２２（図示せず）を前半のＰＭＭ群、ＰＭＭ１０２３およびＰＭＭ１０２４を後半のＰＭＭ群とするＰＭＭ群のペアを形成して、各ペア間を連結し（破線参照）、これらペアを構成する２つのＰＭＭ群の間で、要素の順位番号を確定する。以下、４つのＰＭＭを前半のＰＭＭ群、および、これに引き続く４つのＰＭＭを後半のＰＭＭ群とするＰＭＭ群のペア（一点鎖線参照）、８つのＰＭＭを前半のＰＭＭ群、および、これに引き続く８つのＰＭＭを後半のＰＭＭ群とするＰＭＭ群のペア（点線参照）というように、各々が2^nのＰＭＭ群からなるＰＭＭ群のペアを順次形成し、これらの間で、要素の順位番号を確定する。最終的に、５１２個のＰＭＭを前半のＰＭＭ群、それに引き続く５１２個のＰＭＭを後半のＰＭＭ群とするＰＭＭ群のペアの間で、要素の順位番号を確定することにより、１０２４個のＰＭＭ中の要素すべての順位番号を確定することが可能となる。

【００６５】このように、各々が2^n個のＰＭＭからなるＰＭＭ群のペアを形成して、ペアを構成するＰＭＭ群の各ＰＭＭに格納された要素の順位番号を順次確定することにより（図７のステップ７０９、７１０参照）、最終的に、全ての要素の順位番号が確定すると（ステップ７１０でイエス(Yes)、必要な場合には、上記順位番号付けにしたがった配列を再形成する処理が実行される（ステップ７１１）。この処理は必須ではないが、順位番号にしたがって要素が配置されているような配列を生成することにより、後に実行される情報処理をより高速に実現することが可能となる。

【００６６】より詳細には、まず、ＣＰＵ１２は、各ＰＭＭの入力および出力があるバスに接続されるように、スイッチ２８、３０を制御する。図２１は、ＰＭＭが４つである場合に、これらの間の接続を模式的に示す図である。次いで、ＰＭＭ１４－１～１４－４のＭＰＵは、確定した順位番号にしたがって、要素および順位番号をバス上に放出する。各ＭＰＵは、バス上に放出される要素およびその順位番号をモニターし、もともと自己のＲＡＭコアにて分掌していた要素の添え字（レコード番号）と同一の順位番号を有する要素を取り込み、ＲＡＭコアの所定の領域に格納する。たとえば、添え字（レコード番号）「０」および「１」の要素をもともと自己のＲＡＭコアに記憶していたＰＭＭにおいては（たとえば、図１０のＰＭＭ１４－１参照）、順位番号「０」および「１」を付された要素を取り込み、これらを記憶すれば良い。このようにすれば、各ＰＭＭにおいて、実際にソートされた配列を分掌することが可能となる。なお、このように、ソートされた配列を形成する際にも、ＰＭＭのＭＰＵは、必要な空間ＩＤ管理テーブルを作成する。

【００６７】或いは、図２２に示すように、ソートされた配列を分掌するための他のＰＭＭ（ＰＭＭ１４－５～ＰＭＭ１４－８）を設け、他のＰＭＭ群の各々が、ＰＭＭ１４－１～ＰＭＭ１４－４から、順次出力される要素およびその順位番号をモニターし、順位番号にしたがって、自己が取り込むべき要素を取り込んで、各ＰＭＭのＲＡＭコアに記憶しても良い。たとえば、上記本発明を利用して、１０２４個のＰＭＭを設け、各ＰＭＭに約１００万のデータ（要素）を格納しておき、これらデータのソートを行った場合に、以下のような時間でソートが完了すると考えられる。ここに、各ＰＭＭ間を接続するバスは、６．４ＧＢ／秒のデータ伝送が可能であり、かつ、処理中には、全てのＰＭＭが並列的に動作し（すなわち、処理を実行していないＰＭＭが存在せず）、かつ、関連するすべてのＰＭＭが、同時にかつ協調的に同動作できると仮定する。また、各ＰＭＭにおける約１００万のデータ（要素）のソートは、２．５秒にて完了すると考える。この場合、１０２４個のＰＭＭ中の、約１０億個の要素をソートするために、略４秒程度しか必要としないことがわかる。

【００６８】本実施の形態によれば、各ＰＭＭを、初期的には、２つのＰＭＭのペアに分け、次いで、順次、各群が2^n個のＰＭＭから構成されるＰＭＭ群のペアに分けて、各ペアの間で順位番号を確定させていく。また、各ペアにて利用するバスを、スイッチ等を用いて調整することにより、各ペアにおける順位番号の確定が、並列的に実行することができる。さらに、前半のＰＭＭ群からの要素を、後半のＰＭＭ群に伝達し、後半のＰＭＭ群における構造体配列中の値にしたがって確定させ、確定された順位を、前半のＰＭＭ群に伝達する手順を繰り返すことにより、各ペアにおける順位番号の確定をなすことができる。したがって、処理を実行していない（いわゆる「遊んでいる」）ＰＭＭが生じることなく、極めて並列的に処理を実行できるとともに、バスを利用したデータ転送量を減じることができる。これにより、ソート速度を著しく高速にすることが可能となる。

【００６９】なお、上記第１の実施の形態においては、図１４（ｂ）および図１５（ｂ）に示すようにＰＭＭを接続し、これらの間で各要素に順位番号を付することにソート処理を実現しているが、図１４（ａ）および図１５（ａ）に示すようにＰＭＭを接続しても良い。この場合には、図１３における後半のＰＭＭ群の処理（ステップ１３１２～ステップ１３２３）が並列的に実行されず、あるＰＭＭにおいて仮想順次番号が得られると、処理対象となる要素と当該仮想順位番号が、隣接するＰＭＭに伝達され、当該ＰＭＭにおいて、ステップ１３１２～１３２３の処理が実行される。したがって、後半のＰＭＭ群を構成するＰＭＭの個数が多くなると、それだけ処理の遅延を招く場合もある。

【００７０】［他のソート処理（第２の実施の形態）］

次に、本発明の第２の実施の形態につき説明を加える。上記第１の実施の形態においては、すべての要素（前半のＰＭＭ群内の要素）が、後半のＰＭＭ群に転送されている。しかしながら、配列が巨大になるにしたがって、重複値が多数出現し得る。上記第１の実施の形態にかかる手法では、同じ値をとる要素が何度もバス上に送出される。場合によっては、同じ値の要素を繰り返し送出することは、無駄であると考えることができる。そこで、第２の実施の形態においては、ＰＭＭ群中の要素の個数を予めカウントし、要素とともにその個数を、後半のＰＭＭ群に送出することにより、重複する要素を繰り返しバス上に送出することを防止している。

【００７１】たとえば、４つのＰＭＭの対の各々においてソート処理が終了し、これら対を接続して、８つのＰＭＭにおけるソート処理を実行することを考える。この場合に、図２３に示すように、８つのＰＭＭのマージおよびソート処理を実行するバス（図２３においてＰＭＭの下側に位置するバス、たとえば、符号２３０１～２３０３参照）のほか、他のバス（図２３においてＰＭＭの上側に位置するバス２３０４、２３０５参照）を用いて、ＰＭＭ間のデータの授受ができるのが望ましい。図２３に示すような接続態様において、ＰＭＭ１４－１～ＰＭＭ１４－４（以下、便宜上、「ＰＰＭ１４－１」ないし「ＰＭＭ１４－４」を、それぞれ、「ＰＭＭ１」ないし「ＰＭＭ４」と称する。）における値の重複数を算出する処理につき説明を加える。ここに、ＰＭＭ１～ＰＭＭ４の入出力端子（Ｉ／Ｏ）と接続されたバス（符号２３０４参照）を第１のバスと称し、ＰＭ１～ＰＭＭ４の他の入出力端子（Ｉ／Ｏ）と接続されたバス（符号２３０５参照）を第２のバスと称する。第１のバスは、ＰＭＭ１～ＰＭＭ４からなるＰＭＭ群の情報交換用に用いられ、第２のバスは、値およびその重複数を各ＰＭＭに与えるために利用される。

【００７２】なお、以下の説明においては、図２５に示すように、ＰＭＭ１～ＰＭＭ４中の配列において、各要素に順位番号が付されたものの重複数を算出している。すなわち、重複数は、前半のＰＭＭ群においてのみ算出すれば足りる。図２４は、ＰＭＭ群における重複数を算出するための処理を示すフローチャートである。ＰＭＭ１～ＰＭＭ４の各々は、まず、種々の初期化の処理を実行する（ステップ２４０１）。ここで、各ＰＭＭでは、処理にかかる値（要素）の順位番号を示す順位番号カウンタ、ある値（要素）がどれだけ重複して存在するかを示す同一値個数カウンタ、および、前回の処理において処理対象となった値（要素）を保持する前回値保存レジスタが設けられ、順位番号カウンタおよび同一値個数カウンタの値に初期値「０」が与えられる（図２５参照）。なお、初期的には、前回値保存レジスタには何も値が保持されない。

【００７３】次いで、各ＰＭＭは、順位番号カウンタを

参照して、処理対象となる要素の順位番号を特定し、当該順位番号を付された要素が、自己の掌握するものであるか否かを判断する（ステップ２４０３）。上記例では、初期的に順位番号カウンタのカウンタ値は「０」であるため、ＰＭＭ３が、自己の掌握する要素が処理対象であると判断する（ステップ２４０３でイエス(Yes)）。なお、次のステップ２４０４～２４０５は、最初の処理（すなわち順位番号「０」の要素に関する処理）では無視される。ＰＭＭ３は、順位番号「０」が付された要素（この場合には「あべ」）と同じ要素が、自分の中にいくつ存在するか（すなわち、ＰＭＭ３がいくつ「あべ」という要素を掌握しているか）を判定し、第１のバスに、要素「あべ」と、この要素をいくつ持っているかを示す自己ＰＭＭ内存在数とを送出する（ステップ２４０６）。他のＰＭＭ（ＰＭＭ１、ＰＭＭ２およびＰＭＭ４）では、ステップ２４０３でノー(No)と判断されるため、ステップ２４０７に進む。

【００７４】各ＰＭＭは、第１のバスを介して与えられたデータを受理し、データ中の自己ＰＭＭ内存在数に基づき、順位番号カウンタのカウンタ値に、ＰＭＭ内存在数を加える（ステップ２４０８）。上記例では、順位番号カウンタのカウンタ値が「０＋１＝１」となる。次いで、与えられた要素が、前回値保存レジスタのものと異なるか否かが判断され（ステップ２４０９）、双方が同一であった場合には、同一値個数カウンタのカウンタ値に、自己ＰＭＭ内存在数が加えられ（ステップ２４１０）、その一方、新しい値のときには、後述する入替処理が実行される（ステップ２４１１）。なお、初回の処理では、前回値保存レジスタには何ら値が保持されていないため、上記ステップ２４０９の判断が省略され、かつ、前回値保存レジスタに、要素が収容されるとともに、同一値個数カウンタのカウントアップが実行される。したがって、上記例においては、各ＰＭＭは、受理した要素「あべ」を前回値保存レジスタに記憶するとともに、同一値個数カウンタを「０＋１＝１」とする（図２６参照）。このようなステップ２４０１～２４１１の処理が繰り返され、最後の要素に関する処理が終了すると、ステップ２４０１においてイエス(Yes)と判断され、ステップ２４１２に進む。

【００７５】上記例において、最初のステップ２４０１～２４１１の処理が終了すると、各ＰＭＭは、順位番号カウンタのカウンタ値を参照して、カウンタ値が「１」であることを確認する。これにより、ＰＭＭ４が、順位番号「１」の要素を掌握していることがわかる。また、ＰＭＭ４は、前回値保存レジスタの値（要素「あべ」）と、順位番号「１」が付された要素「あべ」とを比較し（ステップ２４０４）、値に変化がないため、要素「あべ」および自ＰＭＭ内存在数「１」を第１のバスに送出する（ステップ２４０５）。第１のバスを介してデータを受理した各ＰＭＭは、図２７に示すように、順位番号

カウンタをカウントアップ（１＋１＝２）し（ステップ２４０８）、また、前回値保存レジスタに記憶された値と、受理した要素とが同一であるため、同一値個数カウンタをカウントアップ（１＋１＝２）する（ステップ２４１０）。

【００７６】この後に、各ＰＭＭにおいては、順位番号「２」の要素に関する処理が実行される。順位番号「２」の要素の処理では、ＰＭＭ１が要素を保持しているため、ＰＭＭ１が、要素「いとう」と前回値保存レジスタに記憶された要素「あべ」とを比較する。ここでは、値に変化があるため（ステップ２４０４でイエス(Yes)）、ＰＭＭ１は、前回値保存レジスタの内容（要素「あべ」）と、同一値個数カウンタの値「２」を第２のバスに送出する（ステップ２４０５）。このレジスタの内容およびカウンタ値は、各ＰＭＭに与えられる。後述するように、ある要素（この場合には要素「あべ」）の重複数が算出されると、当該要素に関するソート処理（図３１参照）が実行され得る。したがって、各ＰＭＭにおいて、要素およびその重複数は、当該要素に関するソート処理が完了するまで保持していれば良い。また、要素「いとう」および自ＰＭＭ内存在数「１」が第１のバスに与えられる（ステップ２４０６）。

【００７７】各ＰＭＭは、第１のバスを介して与えられたデータに基づき、順位番号カウンタをカウントアップ（２＋１＝３）する（ステップ２４０８）。また、前回値保存レジスタの値「あべ」と、伝達された要素「いとう」とが異なるため（ステップ２４０９でイエス(Yes)）、各ＰＭＭは、前回値保存レジスタの値を書き替える（更新する）とともに、同一値個数カウンタの値を、第１のバスを介して与えられた自ＰＭＭ内存在数に置き換える（ステップ２４１１および図２８（ａ）参照）。

【００７８】他の順位番号の要素についても、同様の処理が施される。たとえば、順位番号「３」に関しては、ＰＭＭ３が、ステップ２４０４、２４０６にしたがって、要素「いとう」を第１のバスに送出し、かつ、各ＰＭＭが、ステップ２４０７、２４０８、２４０９および２４１０にしたがって、各カウンタをカウントアップする（図２８（ｂ）参照）。また、順位番号「４」に関して、ＰＭＭ１が、ステップ２４０４、２４０５、２４０６にしたがって、要素「いとう」および同一値個数カウンタのカウンタ値「２」を第２のバスに送出するとともに、要素「すぎもと」を第１のバスに送出し、かつ、各ＰＭＭが、ステップ２４０７、２４０８、２４０９、２４１１の順で、各カウンタをカウントアップするとともに、レジスタを更新する（図２９（ａ）参照）。

【００７９】順位番号「５」に関する処理では、ＰＭＭ２は、自己が掌握する要素「すぎもと」が２つあることから、第１のバスに、要素「すぎもと」および自ＰＭＭ内存在数「２」を送出する。したがって、各ＰＭＭにおいて、順位番号カウンタ、および、同一値個数カウンタ

のそれぞれのカウンタ値に、「２」が加えられる（図２９（ｂ）参照）。また、この処理により、順位番号カウンタのカウンタ値は、「５」から「７」に変化するため、次の処理対象となる要素の順位番号は「６」ではなく「７」になることに留意されたい。順位番号「７」を付された最後の要素に関する処理（図３０（ａ）参照）が終了すると、ステップ２４０１にてイエス(Yes)と判断される。そこで、先頭のＰＭＭ（上記例では、ＰＭＭ１）は、第２のバスに、要素「すぎもと」および同一値個数カウンタのカウンタ値「４」を送出し（ステップ２４１３）、次いで、第２のバスに処理が終了したことを示すデータを送出する（ステップ２４１４）。各ＰＭＭには、各要素およびその個数を示す存在数が第２のバスを介して与えられ、これがソート処理に利用される。なお、上記例では、先頭のＰＭＭが、ステップ２４１３、２４１４を実行するように構成したが、これに限定されるものではなく、予め、最後の要素等、および、終了を示すデータを出力するＰＭＭを定めておけば良い。上述したように、あるＰＭＭ群における各要素の存在数を得ることにより、ＰＭＭ群を、他のＰＭＭ群とマージして、これらの要素をソートする際に、重複した要素を送る必要がなくなる。

【００８０】図３１は、重複した要素の送出を排除したソート処理を示すフローチャートである。図３１は、一部を除き、図１３の処理と同一であり、その末尾２桁が同じものは、略対応する処理となる。また、図３１において、二重の囲みを付した処理は、新規に追加された処理、或いは、図１３の対応するものと若干異なる処理であることを示している。この処理においては、前半のＰＭＭ群において、処理対象となる要素（すなわち、送出ポインタにより指示される要素）を掌握するＰＭＭは、その要素とともに、前半のＰＭＭ群における当該要素の重複数（存在数）「Ｎ」を、後半のＰＭＭ群に送出する（ステップ３１０３、３１０３－２参照）。たとえば、図２５ないし図３０に示した例において、ＰＭＭ１～ＰＭＭ４からなる前半のＰＭＭ群から、後半のＰＭＭ群に、要素「あべ」が送出される場合には、要素「あべ」のほか、前半のＰＭＭ群における要素「あべ」の重複数「２」が伝達される。また、前半のＰＭＭ群の送出処理において、要素およびその重複数を出力したＰＭＭは、その出力の後に、自己が把握する当該要素の数だけ送出ポインタを移動させる（ステップ３１０４）。たとえば、図２８（ａ）に示すように、要素「あべ」の重複数は「２」であり、これらがＰＭＭ３およびＰＭＭ４において、一つずつ把握されている。したがって、ＰＭＭ３およびＰＭＭ４において、それぞれ、送出ポインタの位置が１つ下方に移動する。なお、各ＰＭＭにおける送出ポインタの移動量の総和が、当該要素の重複数「Ｎ」と等しくなる。

【００８１】その一方、要素およびその重複数を受信した後半のＰＭＭ群を構成する各ＰＭＭにおいては、図３１のステップ３１１６およびステップ３１２０に示すように、前方挿入数および仮想順位番号に、それぞれ、重複数「Ｎ」が加えられる。これは、自分より前方に位置する（順位が小さな）要素が、「Ｎ」だけ存在することに対応している。

【００８２】さらに、前半のＰＭＭ群を構成するＰＭＭの受信処理においては、前半のＰＭＭによる送出処理において送出された要素（比較対象データ）と、後半のＰＭＭによる処理において送出された順位番号とに基づき、受信した順位番号と、比較対象データの送出時における順位番号との差「Ｍ」が算出される（ステップ３１３２－２）。この差「Ｍ」は、後半のＰＭＭ群において、比較対象となっている要素の前方に位置する（すなわち、当該要素より小さい順位番号を付した）要素の数を示している。したがって、後半のＰＭＭ群を構成する各ＰＭＭは、自己の掌握する要素のうち、当該比較対象となっている要素と同一の要素を特定し（ステップ３１３２－３）、存在する場合には、これら要素の順位番号に、それぞれ「Ｍ」を加算する（ステップ３１３４）。ステップ３１３４の後に、ＰＭＭは、当該要素の数だけ受信ポインタを移動させる（ステップ３１３５）。この処理は、ステップ３１０４と略同様である。

【００８３】次に、図２４に示す重複数の算出と、図３１に示すソート処理（場合により、「ソート本体」と称する。）との並列性につき説明を加える。図２３に示すように、本実施の形態においては、重複数のカウントにかかるＰＭＭ間の通信を、バス２３０４、２３０５を利用し、ソート本体の実行にかかるＰＭＭ間の通信を、バス２３０１、２３０２、２３０３等を利用している。そこで、ＰＭＭにおいて並列処理が可能であれば、重複数のカウントとソート本体を並列して実行することができる。この場合に、前半のＰＭＭ群において、ある要素に関する重複数の算出が終了すると（たとえば、図２８（ａ）に示すように、要素「あべ」およびその重複数「２」が第２のバスに送出され、前半のＰＭＭ群を構成するＰＭＭ（ＰＭＭ１～ＰＭＭ４）に受理されると、図３１に示すような処理が、重複数の算出された要素に関して実行可能である。すなわち、ある要素の重複数の算出に応答して、当該要素に関するステップ３１０２～ステップ３１０４の処理、ステップ３１１２～ステップ３１２２の処理、および、ステップ３１３２～ステップ３１３５の処理を実行することができる。また、ある要素に関する要素とその重複数などは、上記図３１に示す処理のうち、当該要素に関するものの終了とともに、削除することが可能である。したがって、前半のＰＭＭ群を構成するＰＭＭの各々において、要素およびその重複数に関するデータ（その量は、異なる要素の数が多くなるに伴って大きくなる）を全て保持しておく必要もない。

【００８４】このように、第２の実施の形態において

は、前半のＰＭＭ群において、重複数を算出し、要素およびその重複数を後半のＰＭＭ群に送出している。したがって、前半のＰＭＭ群が、後半のＰＭＭ群に同一の要素を重複して送る必要がなくなる。特に、同じ要素が数多く重複する場合（たとえば、要素が男女の種別を示すもの、年齢を示すものなど）には、ソート本体の処理回数を減少させることが可能となり、より高速にソート処理を実現することができる。

【００８５】［コンパイル処理（第３の実施の形態）］次に、本発明の第３の実施の形態につき説明を加える。第３の実施の形態では、各ＰＭＭ内に配置された要素からなる配列に基づいて、レコード、各要素を重複なく配置した値リスト、および、レコードから値リストを指定するためのポインタ配列を作成する。この処理を、本明細書においてコンパイルと称する。たとえば、４つのＰＭＭ（ＰＭＭ１～ＰＭＭ４）に、ある配列の要素が分掌されている場合には、図３２に示すように、ＰＭＭを接続すれば良い。図３２に示すように、ＰＭＭ１～ＰＭＭ４の入出力端子（Ｉ／Ｏ）は、第１のバス（符号３２０１参照）により接続され、その一方、ＰＭ１～ＰＭＭ４の出力端子（Ｏ）および他のＰＭＭ“ｋ”の入力端子（Ｉ）は、第２のバス（符号３２０２参照）により接続されている。

【００８６】第１のバスは、ＰＭＭ１～ＰＭＭ４からなるＰＭＭ群の情報交換用に用いられ、第２のバスは、要素およびその重複数を他のＰＭＭ“ｋ”に与えるために利用される。本実施の形態においては、上記要素およびその重複数に基づき、他のＰＭＭ“ｋ”において、値リストおよび存在数配列等が形成される。なお、このＰＭＭ“ｋ”は、ＰＭＭ１～ＰＭＭ４以外のＰＭＭであっても良いが、無論ＰＭＭ１～ＰＭＭ４の何れかであっても良い。図３３は、本実施の形態にかかるコンパイル処理を示すフローチャートである。なお、説明を容易にするために、図３４（ａ）に示すように、ＰＭＭ１～ＰＭＭ４には、要素が分掌されており、これらの間で順位番号を付す処理が既に実行されていると考える。まず、各ＰＭＭにおいて、処理にかかる値（要素）の順位番号を示す順位番号カウンタ、処理の後の当該値（要素）の順位番号を示す値番号カウンタ、当該要素がどれだけ重複して存在するかを示す同一値個数カウンタ、および、前回の処理において処理対象となった値（要素）を保持する前回値保存レジスタが設けられ、各カウンタに初期値「０」が与えられる（ステップ３３０１および図３４（ａ）参照）。なお、初期的には、前回値保存レジスタには値が保持されない。

【００８７】以下、図３３のステップ３３０２～ステップ３３０６の処理は、図２４のステップ２４０２１～２４０６と略同様である。すなわち、各ＰＭＭは、順位番号カウンタを参照して、処理対象となるようその順位番号を特定し、当該順位番号が付された要素が、自己の掌握するものであるか否かを判断する(ステップ３３０３)。図３４の状態では、順位番号カウンタのカウンタ値が「０」であるため、ＰＭＭ３が、第２のバスに、当該ＰＭＭ３が順位番号「０」を付された要素「あべ」をいくつ保持しているかを示す自己ＰＭＭ内存在数（この場合には「１」）を創出する（ステップ３３０６および（図３４（ｂ）参照）。次いで、ＰＭＭ３は、前回値保存レジスタに記憶された要素と、第1のバスに放出した要素とが比較され、これらが相違する場合には、値番号カウンタのカウンタ値を、第1のバスに送出したようその順位番号に代入する（ステップ３３０７）。なお、図３４の状態では、値番号カウンタのカウンタ値が初期値「０」であるため、要素「あべ」にかかる順位番号は変化しない（図３４（ｂ）参照）。

【００８８】次いで、各ＰＭＭにおいては、第1のバスを介して与えられたデータを受理する（ステップ３３０８）。ステップ３３０８～３３１１の処理は、図２４におけるステップ２４０８～２４０１の処理と略同様である。すなわち、各ＰＭＭは、順位番号カウンタのカウンタ値に、与えられたデータのＰＭＭ内存在数を加え、さらに、与えられたデータのうち、要素が新たなものではない場合（ステップ３３１０でノー(No)）には、同一値個数カウンタのカウンタ値に、ＰＭＭ内存在数を加える（ステップ３３１１および図３４（ｂ）参照）。図３４に示すように、順位番号「０」が付された要素「あべ」に関する処理が終了すると、順位番号「１」が付された要素に関する処理が、同様に実行される（図３５（ａ）参照）。

【００８９】さらに、順位番号「２」が付された要素に関する処理が実行される。ここでは、ＰＭＭ１が、前回値保存レジスタに記憶された要素「あべ」と、順位番号「２」が付された要素「いとう」とを比較する。ここでは、これらが相違しているため（ステップ３３０４でイエス(Yes)）、ＰＭＭ１は、第２のバスに、前回値保存レジスタに記憶された要素と、同一値個数カウンタのカウンタ値とを送出する（ステップ３０５）。次いで、ＰＭＭ１は、第1のバスに、処理対象の要素「いとう」と、ＰＭＭ１が掌握する要素「いとう」の数である自ＰＭＭ内存在数「１」とを送出する（ステップ３３０６）。その後に、ＰＭＭ１は、前回値保存レジスタに記憶された要素と、第1のバスに放出した要素とを比較する。要素「いとう」を放出する場合には、これらが相違するため、要素「いとう」の順位番号に、値番号カウンタのカウンタ値に「１」を加えた値（０＋１＝１）を代入する。各ＰＭＭは、第1のバスを介して与えられたデータを受理し（ステップ３３０８）、順位番号カウンタのカウンタ値に、受け入れたデータ中の自ＰＭＭ内存在数を加算する（２＋１＝３）（ステップ３３０９および図３５（ｂ）参照）。要素「いとう」が与えられた場合には、前回値保存レジスタの要素「あべ」と与えられた

要素「いとう」とが異なるため（ステップ３３１０でイエス(Yes)）、各ＰＭＭは新値登録処理を実行する（ステップ３３１２）。この処理においては、値番号カウンタのカウンタ値がインクリメント（０＋１＝１）され、同一値個数カウンタのカウンタ値が、受理したデータ中の自ＰＭＭ内存在数「１」に変更され、かつ、前回値保存レジスタの内容が、要素「いとう」に書きかえられる（図３５（ｂ）参照）。

【００９０】順位番号「３」の要素「いとう」に関しても同様の処理が施される。たとえば、ＰＭＭ３は、第1のバスに要素「いとう」および自ＰＭＭ内存在数「１」を送出し（ステップ３３０６参照）、かつ、当該要素「いとう」の順位番号に、値番号カウンタのカウンタ値「１」を代入する（ステップ３３０７および図３６（ａ）参照）。また、各ＰＭＭは、順位番号カウンタのカウンタ値に、受理した自ＰＭＭ内存在数「１」を加える（ステップ３３０９参照）とともに、同一値個数カウンタのカウンタ値に、自ＰＭＭ内存在数「１」を加える（ステップ３３１１および図３６（ａ）参照）。さらに、順位番号「４」の要素「すぎもと」に関しても、図３６（ｂ）に示すように、ＰＭＭ１が、第２のバスに、要素「いとう」および同一値個数カウンタのカウンタ値「２」を送出し（ステップ３３０５）、第1のバスに、要素「すぎもと」および自ＰＭＭ内存在数「１」を送出し（ステップ３３０６）、かつ、要素「すぎもと」の順位番号に、値番号カウンタに「１」を加えた値（１＋１＝２）を代入する。その一方、各ＰＭＭにおいても、順位番号カウンタのカウントアップおよび新値登録処理が実行される（ステップ３３０９、３３１２および図３６（ｂ）参照）。他の順位番号の要素についても、同様の処理が施される。各要素についての処理は、図３７（ａ）、（ｂ）および図３８に示されている。なお、図３８に関して、ＰＭＭ１は、最後の要素「すぎもと」、および、その要素の存在数を第1のバスに送出し、かつ、終了を示すデータを第２のバスに出力する（ステップ３３１５参照）。

【００９１】前述したように、第２のバスには、ＰＭＭ“ｋ”の入力が接続されている。したがって、第２のバスには、重複のない要素と、これに関する値番号カウンタのカウンタ値とが与えられる。したがって、ＰＭＭ“ｋ”はこれらを受理し、受理した要素を、値リストに順次配置するとともに、受理した値番号カウンタのカウント値を、存在数配列に順次配置する。図３９（ａ）は、ＰＭＭ“ｋ”内に作成された値リストおよび存在数配列を示す図である。これらは、ステップ３３０５或いはステップ３３１４にて送出され（図３５（ｂ）、図３６（ｂ）および図３８参照）、ＰＭＭ“ｋ”に伝達されている。図３９（ａ）に示すように、要素が重複なく値リストに配置され、かつ、各要素がいくつ存在するかを示す存在数（すなわち重複数）が存在数配列に配置されている。

【００９２】さらに、ＰＭＭ１～ＰＭＭ４において、レコードと、各要素に、重複がないように付された順位番号とを対応させる値リストへのポインタ配列を作成することができる。つまり、レコードと、当該レコードに対応する要素に付された順位番号とを対応させた配列を作成すれば、これが値リストへのポインタ配列とすることができる（図３９（ｂ）参照）。図３９（ｂ）において、レコード「０」に関して、対応する要素の順位番号「２」が、値リストへのポインタ配列におけるポインタ値となる。これは、値リスト（図３９（ａ）参照）において、格納位置番号「２」であるような値を指示すべきことを示している。すなわち、値リストへのポインタ配列のポインタ値により、ＰＭＭ“ｋ”に格納された値リストを指示することができ、これにより、レコードから要素を特定することが可能となる。

【００９３】このように本実施の形態によれば、ＰＭＭに分掌された配列の要素をソートして順位番号を付し、かつ、同一の要素には同一の順位を付すように、順位を振り直している。要素は、新たに得られた重複のない順位と対応付けられて、値リストに格納される。当該順位は、値リストへのポインタ配列として、分掌された配列中の要素に対応つけられている。したがって、レコードに基づき、ポインタ配列のポインタ値を経て、値リスト中の要素を特定することが可能となる。

【００９４】［値リストの共有化（第４の実施の形態）］次に、本発明の第４の実施の形態につき説明を加える。第４の実施の形態においては、二つの配列を共有化（ジョイン）している。この前提として、コンパイル処理による値リスト、および、値リストへのポインタ配列が作成されている。また、値リストや、値リストへのポインタ配列には、空間ＩＤが付され、各ＰＭＭは、当該空間ＩＤ等により、自己が分掌している配列に関する種々の情報を把握している。

【００９５】図４０は、第４の実施の形態にかかる共有化処理を示すフローチャートである。説明を容易にするために、図４１（ａ）に示すように、元のデータとして、レコードに対応した要素からなる配列（符号４１００参照）が、あるＰＭＭ群に分掌されていると考える。このレコード群に関するコンパイル処理により、ＰＭＭ１およびＰＭＭ２からなるＰＭＭ群に、ポインタ配列（符号４１０１参照）および値リスト（符号４１０２参照）からなるブロック（以下、「情報ブロック」と称する。）が形成されている。その一方、元のデータとして、レコードに対応する要素からなる他の配列（符号４１１０参照）が、他のＰＭＭ群に分掌され、かつ、コンパイル処理により、ＰＭＭ群３およびＰＭＭ群４からなるＰＭＭ群に、ポインタ配列（符号４１１１参照）および値リスト（符号４１１２参照）からなる情報ブロックが形成されていると考える。

【００９６】各ＰＭＭに、ＣＰＵ１２から、値リストのジョインを指示するインストラクションが、二つの値リストを示す配列の空間ＩＤとともに伝達される。各ＰＭＭのうち、ジョインすべき配列が自己の掌握する値リスト或いはその部分であるようなもの（つまり、上記例では、ＰＭＭ１～ＰＭＭ４）は、空間ＩＤに基づき、ジョインの対象となる値リストを特定する（ステップ４００１および図４２（ａ）参照）。次いで、ＰＭＭ１～ＰＭＭ４は、二つの値リストを合併した状態で、これらをソートして各要素に順位番号を付与する（ステップ４００２）。このソート処理のために、第１の実施の形態にかかるソート処理を利用することができる。上記例においては、まず、ＰＭＭ１およびＰＭＭ２からなる第１のＰＭＭ群、および、ＰＭＭ３およびＰＭＭ４からなる第２のＰＭＭ群のそれぞれにおいて、要素の順位番号付与の処理を実行し、次いで、第１のＰＭＭ群を前半のＰＭＭ群とし、かつ、第２のＰＭＭ群を後半のＰＭＭ群とすることで、二つのＰＭＭ群中の要素に順位番号を付与する。図４２（ｂ）は、このようにして要素に順位番号が付された状態を示す図である。

【００９７】その後に、処理対象となる値リストを分掌するＰＭＭの間でコンパイル処理が実行され、これにより、他のＰＭＭ或いはＰＭＭ１～ＰＭＭ４の何れかに、共通化された値リストおよび共通化された存在数配列が生成される（ステップ４００３）。すなわち、コンパイル処理により、マージされた値リストの要素が重複しないような新たな値リストと、各要素がどれだけ重複して存在しているかを示す存在数を格納した存在数配列が得られる（図４２（ｃ）参照）。このような処理の後に、ジョインされた新たな値リスト（すなわちコンパイル処理により得られた値リスト）を指示するための新たなポインタ配列が求められる。これは、共有化前の情報ブロックにおけるポインタ配列中のポインタ値が示す、コンパイル処理により得られた順位番号配列の対応する順位番号を、当該ポインタ配列中のポインタ値の位置と対応する位置に格納するような新たなポインタ配列を作成することにより実現される。上記順位番号配列中の値は、各要素に付された新たな順位番号（図４２（ｃ）参照）に対応することは理解できるであろう。

【００９８】図４３（ａ）に示すように、たとえば、ポインタ配列中の第１のポインタ値「１」の示す位置の順位番号配列中の値（順位番号）は「２」であるため、共有化された後のポインタ配列中の対応する位置のポインタ値は「２」となる。また、第２のポインタ値「２」の示す位置の順位番号配列中の値（順位番号）は「３」であるため、共有化されたポインタ配列中の対応する位置のポインタ値は「３」となる。このようにして、ジョインされた値リストに関するポインタ配列を得ることが可能となる（図４３（ａ）および図４３（ｂ））。

【００９９】このような、新たなポインタ配列、および、ジョインされた値リストにより、レコードから値（要素）を特定できることは明らかであろう。図４４に示すように、レコードが、新たに得られた値リストへのポインタ配列中、対応する位置のポインタ値を特定し、かつ、当該ポインタ値が、その値が示す位置にある、値リスト中の要素を特定する。ここで、二つの値リストがジョインされているにもかかわらず、元のデータの要素と同一の要素が指定されることが理解できるであろう。

【０１００】このように、第４の実施の形態によれば、複数の値リストを併合して、併合された値リストの要素に関して、ソート処理とコンパイル処理とを組み合わせることにより、ジョインされた値リスト、および、各値リストの順位番号配列を得る。レコードから値リストを指定するためのポインタ配列により順位番号配列の値（順位番号）が特定され、当該順位番号を、レコードに対応する位置に格納することにより、レコードに基づきジョインされた値リストを指定するための、新たなポインタ配列を得ることができる。したがって、上述したソート処理の時間およびコンパイル処理の時間程度で、複数の値リストをジョインすることが可能となり、著しく処理速度を向上させることが可能となる。

【０１０１】本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。たとえば、前記実施の形態においては、本発明を、コンピュータシステムに適用しているがこれに限定されるものではなく、パーソナルコンピュータなどに接続可能なコンピュータボードに適用することもできる。この場合には、図１において、ＣＰＵ１２、メモリユニット１４、バス２４等がボード上に搭載され、これが、本発明における情報処理ユニットを構成する。

【０１０２】また、ＣＰＵ１２とメモリモジュール１４との間、および／または、メモリモジュール１４間を接続するバスの組の数は、前記実施の形態に限定されるものではなく、コンピュータシステムを搭載する回路基板の大きさ、各バスのビット数などを考慮して適宜決定することができる。また、前記実施の形態においては、メモリモジュールの入出力とバスとの接続を規定するためのスイッチ２８と、ＣＰＵとメモリモジュールとの間、メモリモジュール間、或いは、メモリモジュールの入出力間で、バスの切断することができるスイッチ３０とを設けている。スイッチ２９、３０を設けることにより、たとえば、あるバス（図１のバス２４－４参照）を、ＣＰＵモジュール１２とメモリモジュール１４－１とのデータ授受のために利用するとともに、同時に、メモリモジュール１４－２とメモリモジュール１４－３との間のデータ授受のために利用することができる（この場合に、スイッチ３０－５をオフにすれば良い）。したがって、より有効にバスを利用することが可能となってい

る。しかしながら、バスの組を数を十分に大きくできる場合、或いは、メモリモジュールの数が比較的少ない場合には、スイッチ２９或いは３０を必ずしも設けなくて良い。

【０１０３】また、本明細書において、制御信号ライン２５を介して、ＣＰＵ１２からのインストラクションが与えられることを記載したが、制御信号ライン２５を介して、インストラクションのほか、クロックなど、各メモリモジュールが同期して作動するための種々の制御信号が与えられ、かつ、各メモリモジュールからＣＰＵ１２への所定の信号（たとえば、エラー信号や、データ受理を示す信号）が与えられていることは言うまでもない。

【０１０４】さらに、前記実施の形態において、ＰＭＭ間の種々の接続を例示したが、ＰＭＭ間の接続や送受信に利用するバスの選択は、上記実施の形態に示すものに限定されない。

【０１０５】また、前記第３の実施の形態においては、図３２に示すように、第１のバス（符号３２０１）を利用して、各ＰＭＭ間の通信をなし、かつ、第２のバス（符号３２０２）参照）を利用して、要素や当該要素の存在数（重複数）が通信されているがこれに限定されるものではなく、たとえば、図４５に示すように、重複のない要素の配列である値リストやその存在数配列を生成するＰＭＭ“ｋ”が、第１のバス４５０１をモニターして、第１のバス４５０１上に表れる要素や存在数配列に基づき、所定の処理（たとえば、ＰＭＭ１～ＰＭＭ４にて実行されるカウンタのカウントアップやレジスタの内容の保持／更新）を実行して、値リストや存在数配列を作成しても良い。

【０１０６】さらに、本明細書において、一つの手段の機能が、二つ以上の物理的手段により実現されても、若しくは、二つ以上の手段の機能が、一つの物理的手段により実現されてもよい。

【０１０７】

【発明の効果】本発明によれば、著しく高速に、かつ、安定した処理時間で、配列のソート、コンパイルおよびジョインが可能な情報処理装置を提供することが可能となる。

【図面の簡単な説明】

【図１】　図１は、本発明の実施の形態にかかるコンピュータシステムの構成を示すブロックダイヤグラムである。

【図２】　図２は、本実施の形態にかかるメモリモジュールの概略を示すブロックダイヤグラムである。

【図３】　図３は、本実施の形態にかかるメモリモジュール間のパイプライン処理を説明するための図である。

【図４】　図４は、本実施の形態にかかる多空間メモリの下での、メモリモジュール１４の構造を説明するための図である。

【図５】　図５は、本実施の形態におけるメモリモジュールへのアクセスを説明するための図である。

【図６】　図６は、第１の実施の形態にかかるソート処理を施す配列の一例を示す図である。

【図７】　図７は、第１の実施の形態にかかるソート処理の処理手順を示すフローチャートである。

【図８】　図８は、第１の実施の形態にかかるソート処理を実施する際のメモリモジュール間の接続を示すブロックダイヤグラムである。

【図９】　図９は、図８に示すメモリモジュール間の接続を模式的に示す図である。

【図１０】　図１０は、第１の実施の形態にかかるソート処理における配列中の要素への番号付与を示す図である。

【図１１】　図１１は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１２】　図１２は、第１の実施の形態にかかるソート処理における配列中の要素への番号付与を示す図である。

【図１３】　図１３は、第１の実施の形態にかかるメモリモジュールのペア間の順位番号処理を示すフローチャートである。

【図１４】　図１４は、図８に示すメモリモジュールに関する、２つのメモリモジュール群の接続例を示すブロックダイヤグラムである。

【図１５】　図１５は、図１４に示す接続例を模式的に示す図である。

【図１６】　図１６は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１７】　図１７は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１８】　図１８は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図１９】　図１９は、第１の実施の形態にかかるソート処理における配列中の要素への順位番号付与を示す図である。

【図２０】　図２０は、第１の実施の形態かかるソート処理におけるメモリモジュールの組み合わせを説明するための図である。

【図２１】　図２１は、第１の実施の形態にかかるソート処理の結果得られた順位番号にしたがって、新たな配列を生成する場合のメモリモジュールの接続例を示す図である。

【図２２】　図２２は、第１の実施の形態にかかるソート処理の結果得られた順位番号にしたがって、新たな配列を生成する場合のメモリモジュールの他の接続例を示

41

す図である。

【図２３】　図２３は、第２の実施の形態にかかるソート処理におけるメモリモジュールの接続例を模式的に示す図である。

【図２４】　図２４は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２５】　図２５は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２６】　図２６は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を示すフローチャートである。

【図２７】　図２７は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２８】　図２８は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図２９】　図２９は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図３０】　図３０は、第２の実施の形態にかかるメモリモジュール群における重複数を算出するための処理を説明するための図である。

【図３１】　図３１は、重複した要素の送出を排除したソート処理を示すフローチャートである。

【図３２】　図３２は、本発明の第３の実施の形態にかかるコンパイル処理におけるメモリモジュールの接続例を模式的に示す図である。

【図３３】　図３３は、第３の実施の形態にかかるコンパイル処理を示すフローチャートである。

【図３４】　図３４は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３５】　図３５は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３６】　図３６は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

＊

42

＊【図３７】　図３７は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３８】　図３８は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図３９】　図３９は、第３の実施の形態にかかるメモリモジュール群におけるコンパイル処理を説明するための図である。

【図４０】　図４０は、本発明の第４の実施の形態にかかる共有化処理を示すフローチャートである。

【図４１】　図４１は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

【図４２】　図４２は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

【図４３】　図４３は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

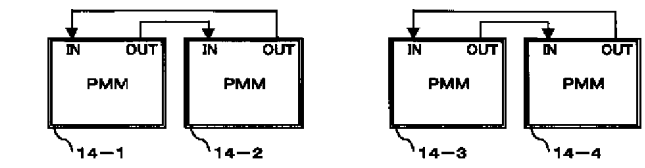
【図４４】　図４４は、第４の実施の形態にかかるメモリモジュール群におけるジョイン処理を説明するための図である。

【図４５】　図４５は、本発明の他の応用例におけるメモリモジュール群の接続を概略的に示す図である。

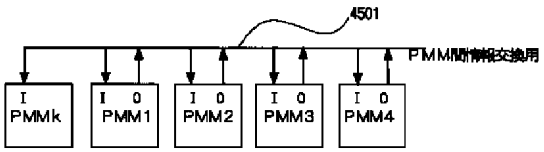
【符号の説明】

１０	コンピュータシステム
１２	ＣＰＵモジュール
１４	メモリモジュール
１６	固定記憶装置
１８	入力装置
２０	表示装置
２２	レガシーメモリ
２４、２６	バス
２５	制御信号ライン
２８、２９、３０	スイッチ
３２	クロックバッファ
３４	ＲＡＭコア
３６	ＭＰＵ
３８	Ｉ／Ｏ

【図９】



【図４５】



【図１】

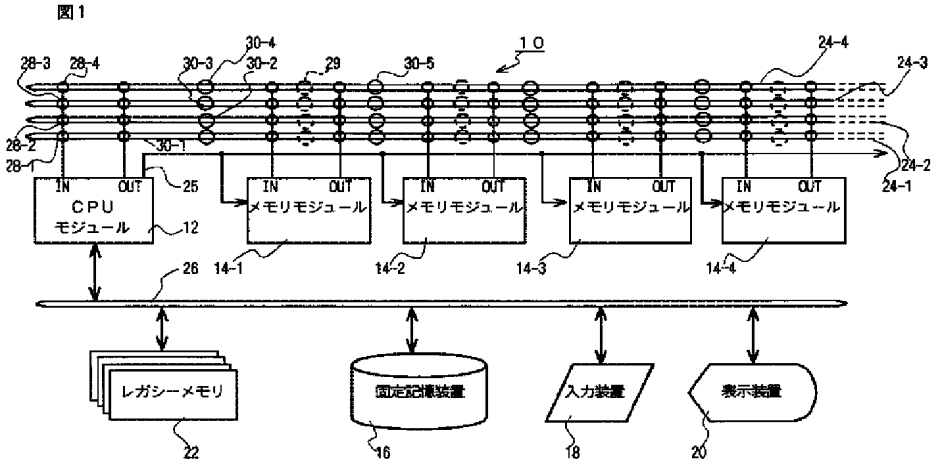


図1
10
CPU
モジュール
12
メモリモジュール
14-1
14-2
14-3
14-4
IN
OUT
25
26
24-1
24-2
24-3
24-4
28-1
28-2
28-3
28-4
29
30-1
30-2
30-3
30-4
30-5
レガシーメモリ
22
固定記憶装置
16
入力装置
18
表示装置
20

【図２】

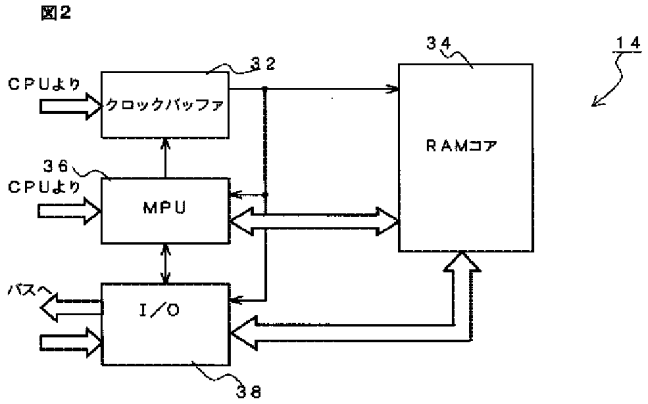


図2
CPUより
クロックバッファ
32
34
RAMコア
14
CPUより
36
MPU
バスへ
I／O
38

【図３】

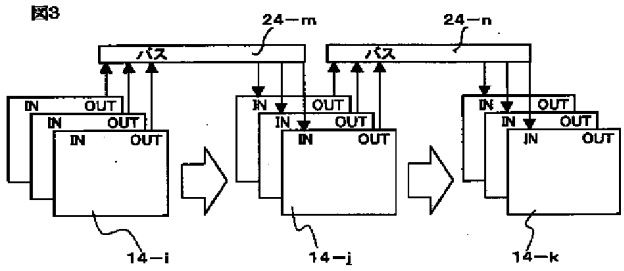


図3
24－m
24－n
バス
IN
OUT
14－i
14－j
14－k

【図４】

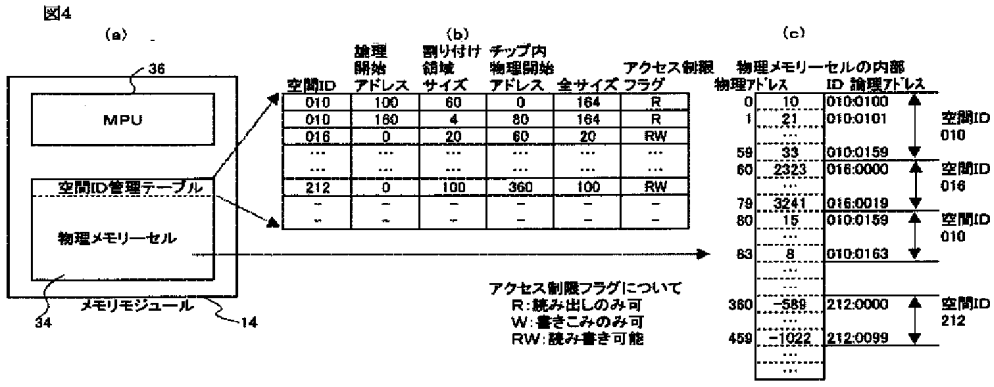
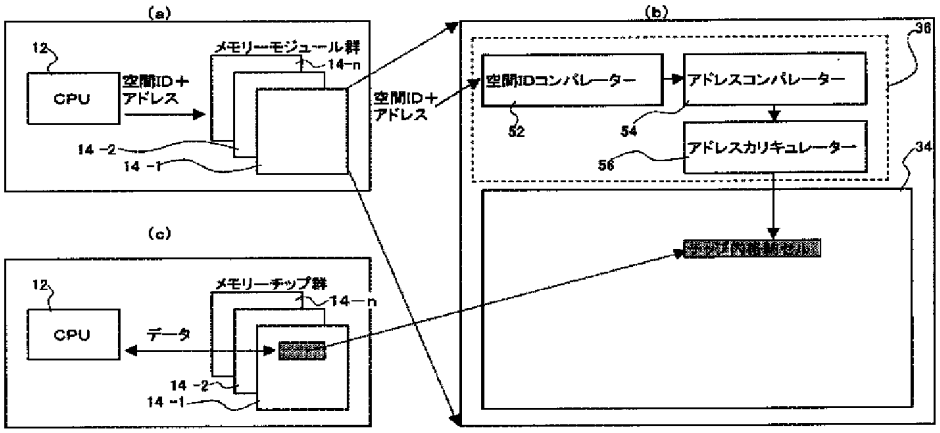


図4
(a)
36
MPU
空間ID管理テーブル
物理メモリーセル
メモリモジュール
34
14
(b)
空間ID
論理開始アドレス
割り付け領域サイズ
チップ内物理開始アドレス
全サイズ
アクセス制限フラグ
010 100 60 0 164 R
010 160 4 80 164 R
016 0 20 60 20 RW
212 0 100 360 100 RW
アクセス制限フラグについて
R:読み出しのみ可
W:書きこみのみ可
RW:読み書き可能
(c)
物理メモリーセルの内部ID
物理アドレス
論理アドレス
0 10 010:0100
1 21 010:0101
59 33 010:0159
60 2323 016:0000
79 3241 016:0019
80 15 010:0159
83 8 010:0163
360 −588 212:0000
459 −1022 212:0099
空間ID 010
空間ID 016
空間ID 010
空間ID 212

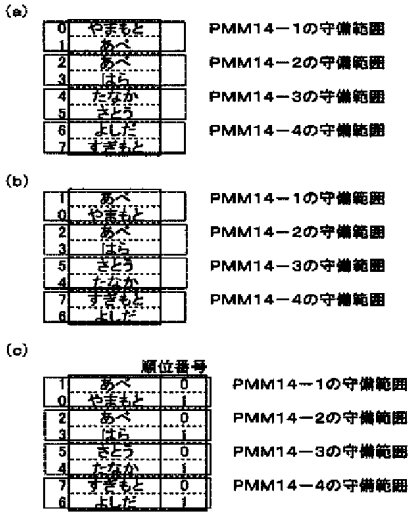
【図５】

図5



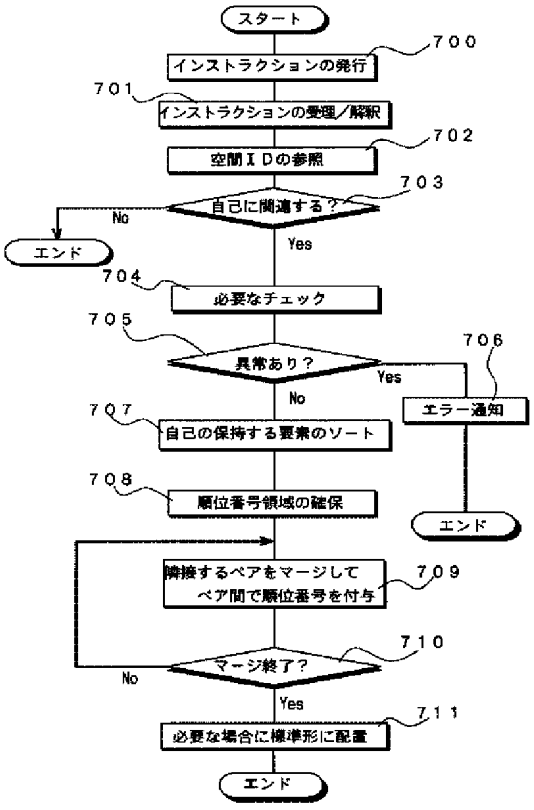
【図６】

図6



【図７】

図7



【図８】

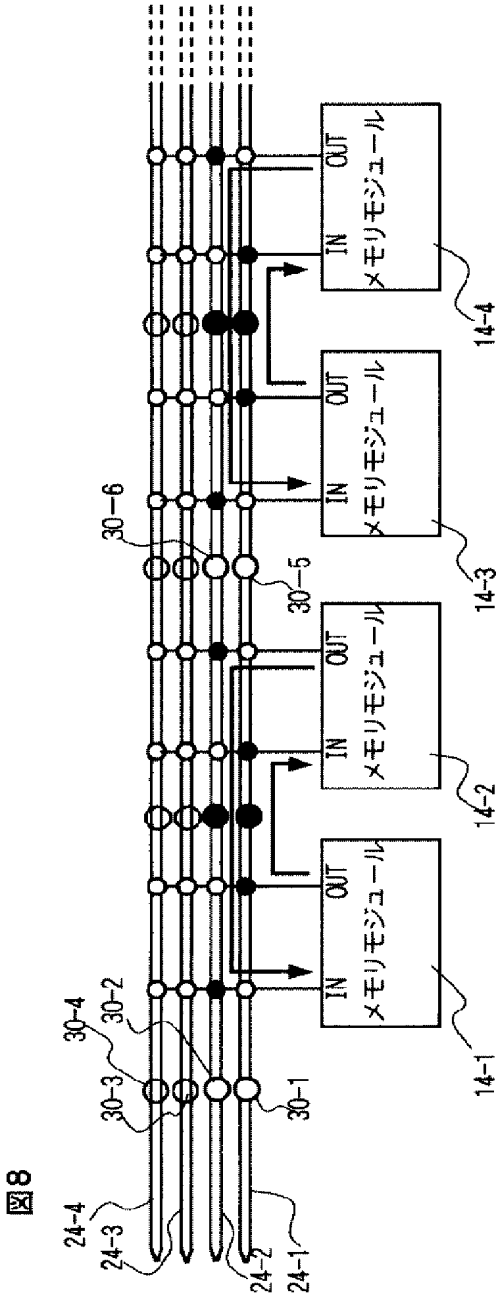
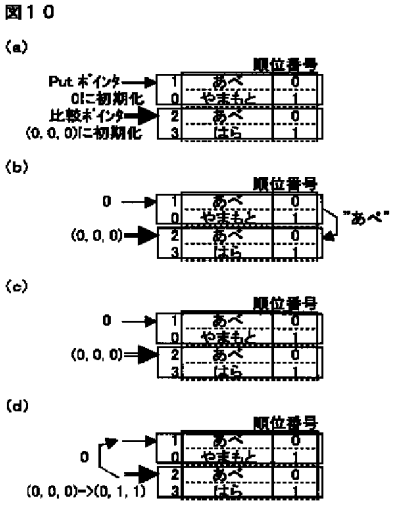
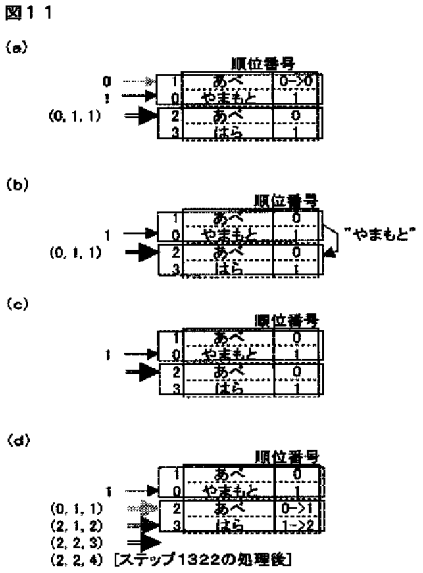


図８

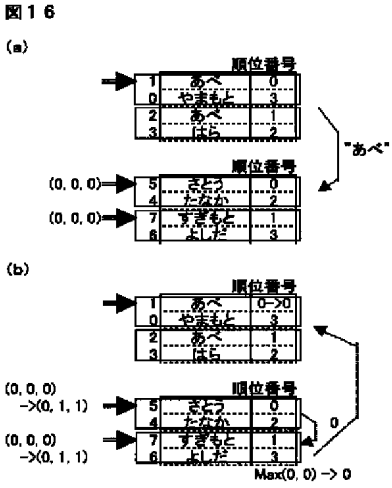
【図１０】



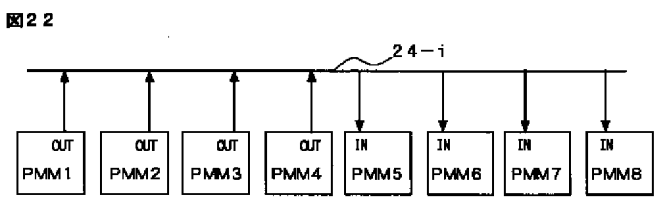
【図１１】



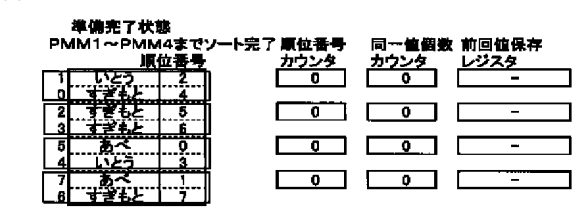
【図１６】



【図２２】

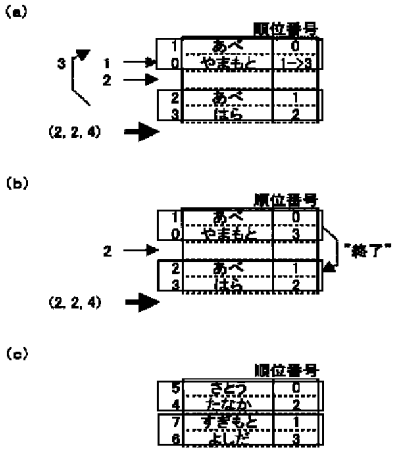


【図２５】



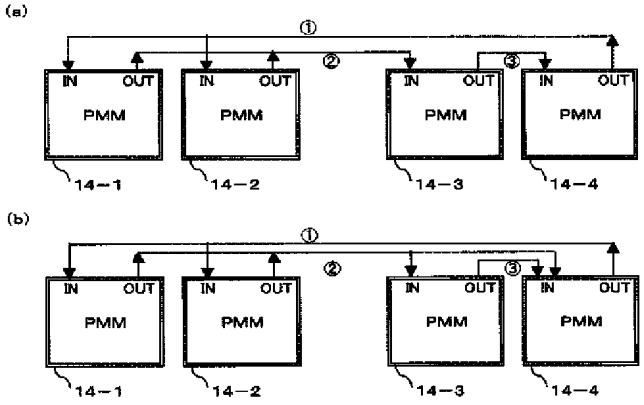
【図１２】

図１２



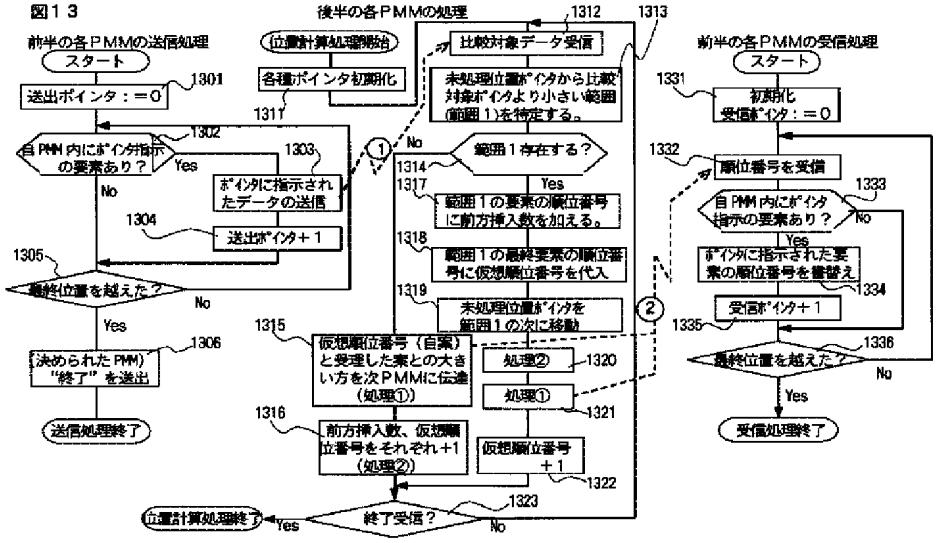
【図１５】

図１５



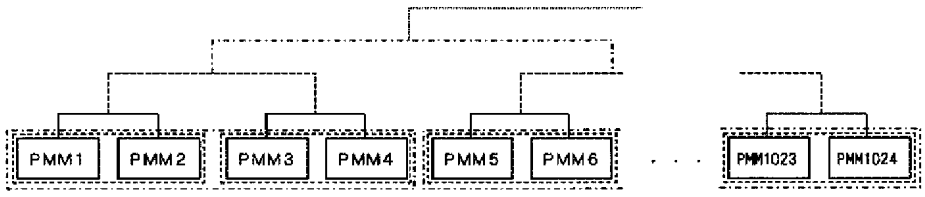
【図１３】

図１３



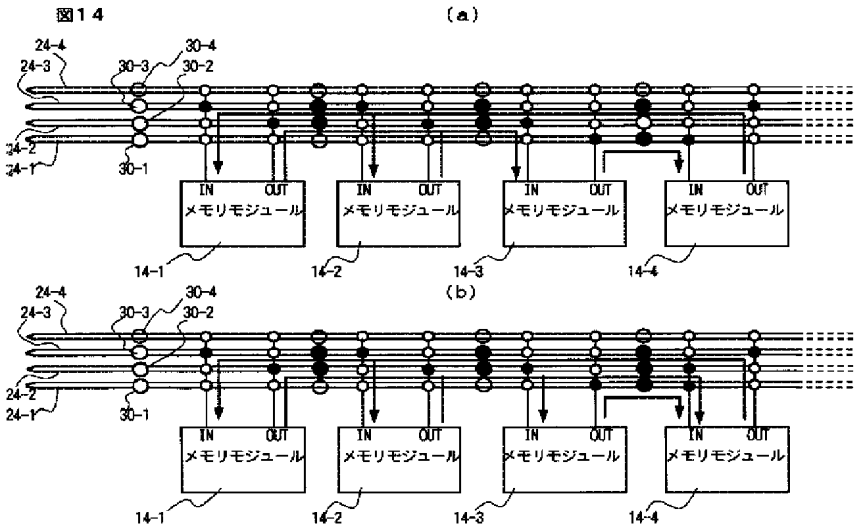
【図２０】

図２０



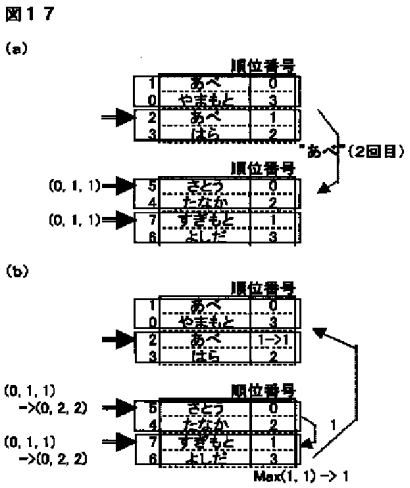
【図１４】

図１４



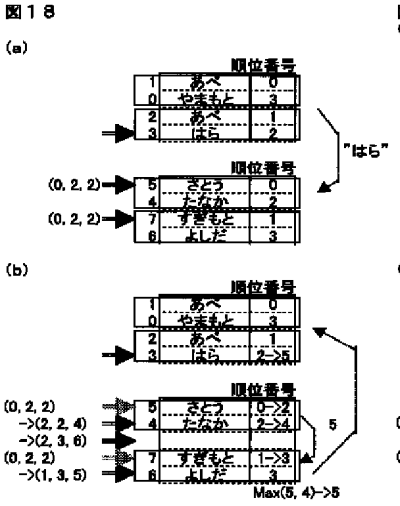
【図１７】

図１７



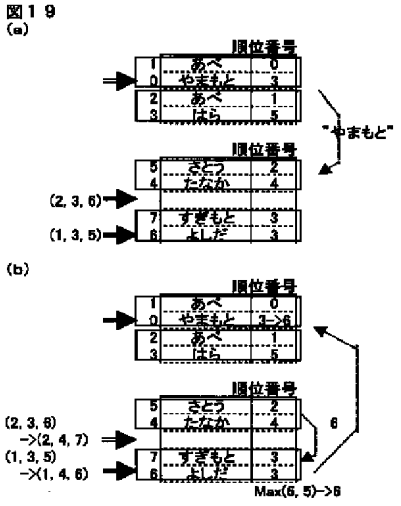
【図１８】

図１８



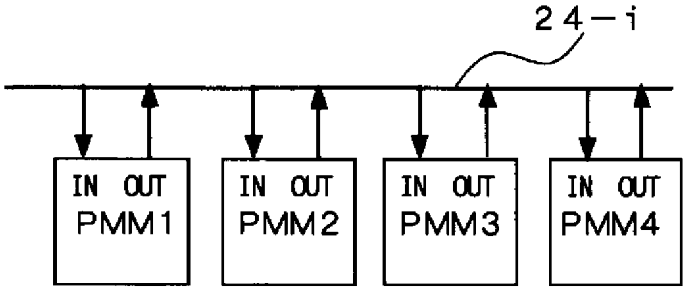
【図１９】

図１９



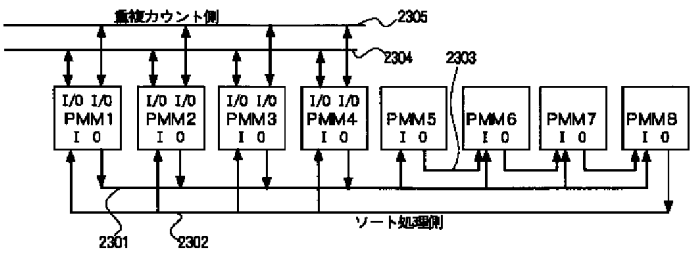
【図２１】

図２１



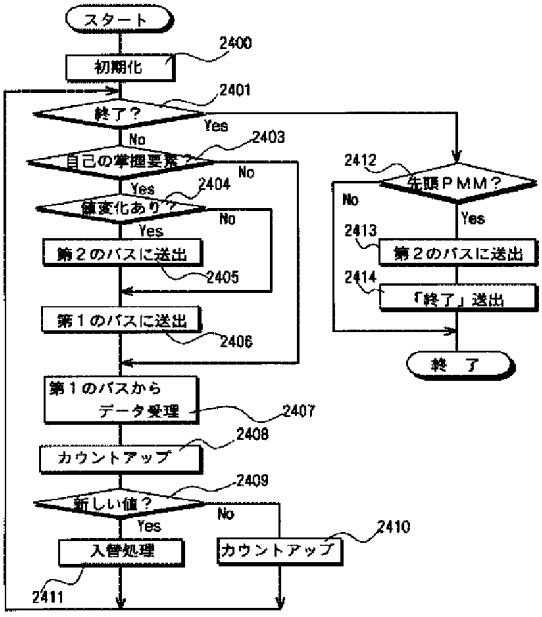
【図２３】

図２３



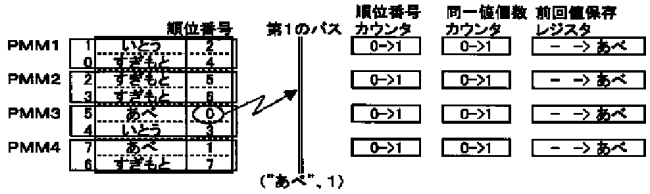
【図２４】

図２４



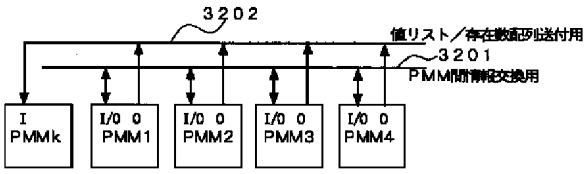
【図２６】

図２６



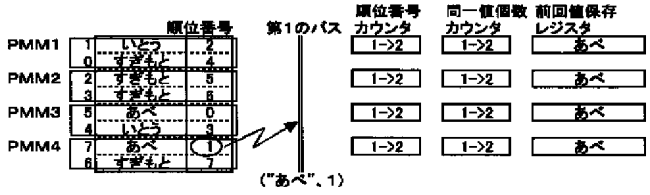
【図３２】

図３２



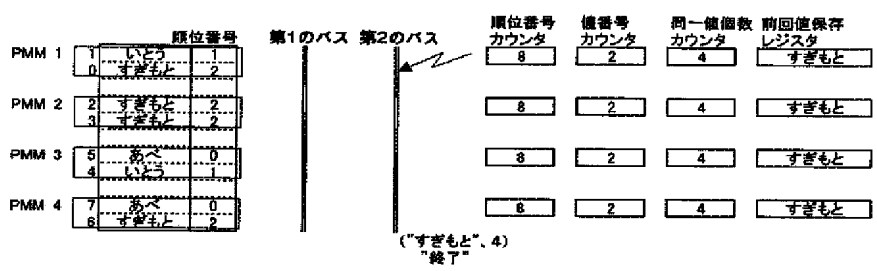
【図２７】

図２７



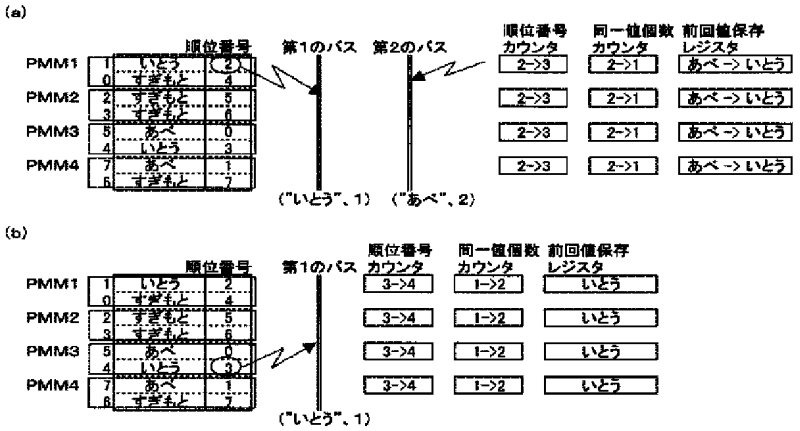
【図３８】

図３８



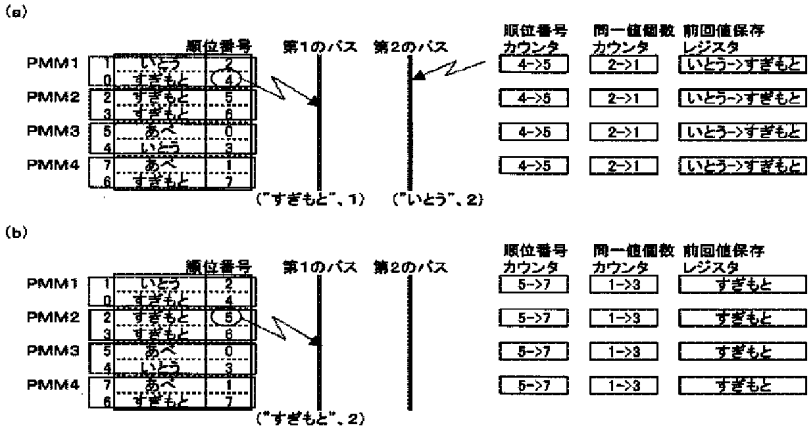
【図２８】

図２８



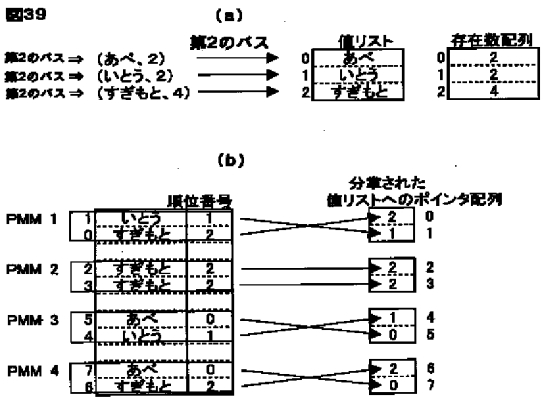
【図２９】

図２９



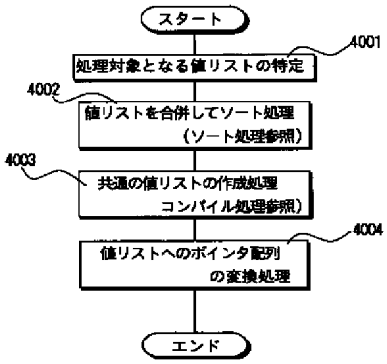
【図３９】

図39



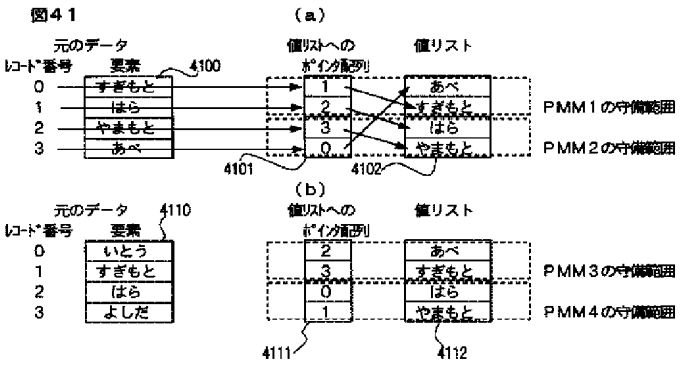
【図４０】

図４０



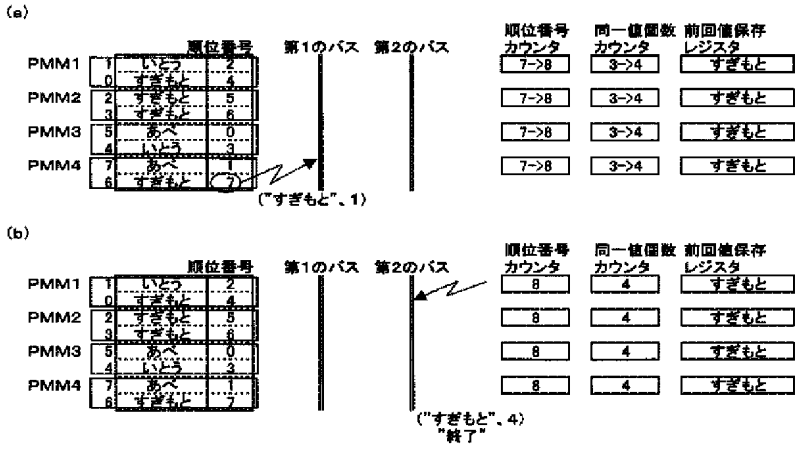
【図４１】

図４１



【図３０】

図３０


(a)
順位番号
第1のパス
第2のパス
順位番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
PMM1
PMM2
PMM3
PMM4
7->8
3->4
すぎもと
("すぎもと"、1)
(b)
8
4
("すぎもと"、4)
"終了"

【図３１】

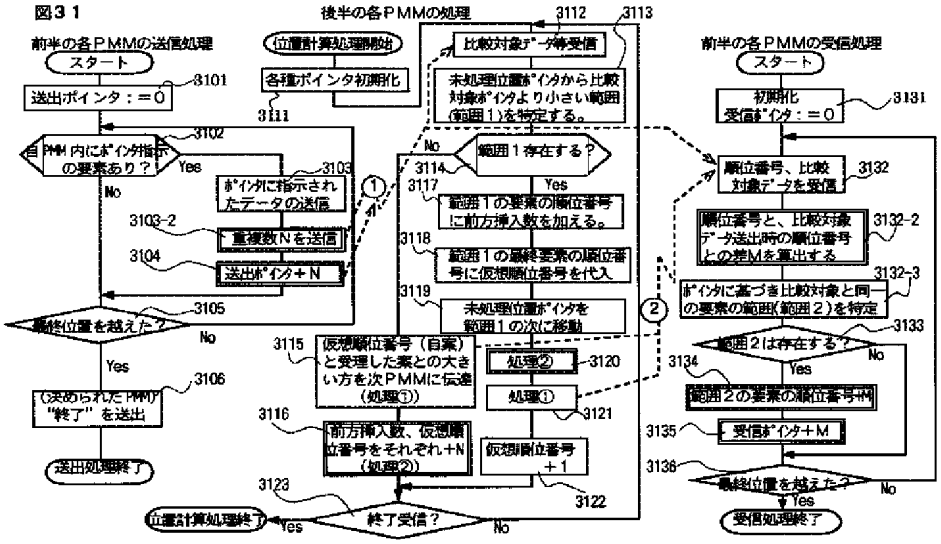


図３１
前半の各ＰＭＭの送信処理
後半の各ＰＭＭの処理
前半の各ＰＭＭの受信処理
スタート
位置計算処理開始
送出ポインタ：＝０
各種ポインタ初期化
比較対象データ等受信
未処理位置ポインタから比較対象ポインタより小さい範囲（範囲１）を特定する。
初期化
受信ポインタ：＝０
自PMM内にポインタ指示の要素あり？
ポインタに指示されたデータの送信
重複数Ｎを送信
送出ポインタ＋Ｎ
範囲１存在する？
範囲１の要素の順位番号に前方挿入数を加える。
範囲１の最終要素の順位番号に仮想順位番号を代入
未処理位置ポインタを範囲１の次に移動
順位番号、比較対象データを受信
順位番号と、比較対象データ送出時の順位番号との差Ｍを算出する
ポインタに基づき比較対象と同一の要素の範囲(範囲２)を特定
範囲２は存在する？
範囲２の要素の順位番号＋Ｍ
受信ポインタ＋Ｍ
最終位置を越えた？
仮想順位番号（自案）と受理した案との大きい方を次ＰＭＭに伝達（処理③）
処理②
処理①
前方挿入数、仮想順位番号をそれぞれ＋Ｎ（処理②）
仮想順位番号＋１
終了受信？
(決められたPMM)"終了"を送出
送出処理終了
位置計算処理終了
受信処理終了
Yes
No

【図３３】

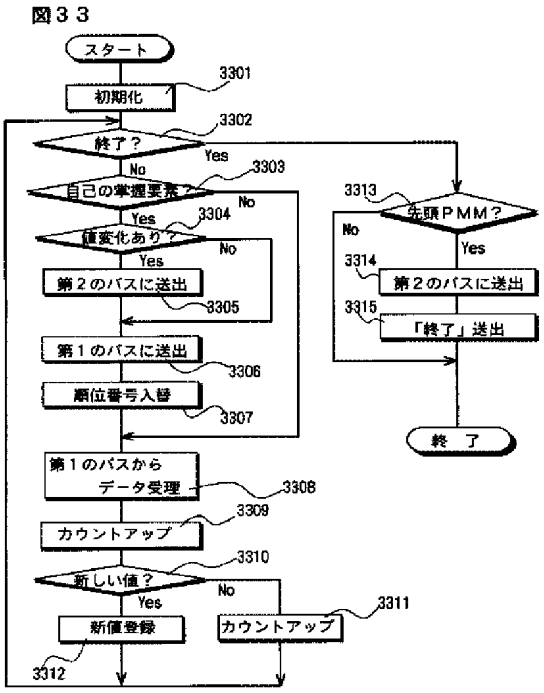


図３３
スタート
初期化
3301
終了？
3302
Yes
No
自己の掌握要素？
3303
No
Yes
値変化あり？
3304
No
Yes
第２のパスに送出
3305
第１のパスに送出
3306
順位番号入替
3307
第１のパスから
データ受理
3308
カウントアップ
3309
新しい値？
3310
No
Yes
新値登録
3312
カウントアップ
3311
先頭ＰＭＭ？
3313
No
Yes
第２のパスに送出
3314
「終了」送出
3315
終　了

【図４２】

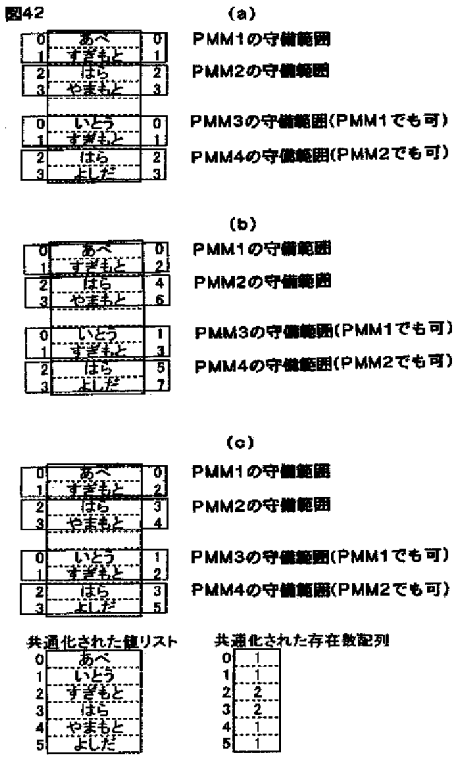


図42
(a)
PMM1の守備範囲
PMM2の守備範囲
PMM3の守備範囲(PMM1でも可)
PMM4の守備範囲(PMM2でも可)
(b)
PMM1の守備範囲
PMM2の守備範囲
PMM3の守備範囲(PMM1でも可)
PMM4の守備範囲(PMM2でも可)
(c)
PMM1の守備範囲
PMM2の守備範囲
PMM3の守備範囲(PMM1でも可)
PMM4の守備範囲(PMM2でも可)
共通化された値リスト
共通化された存在数配列

【図３４】

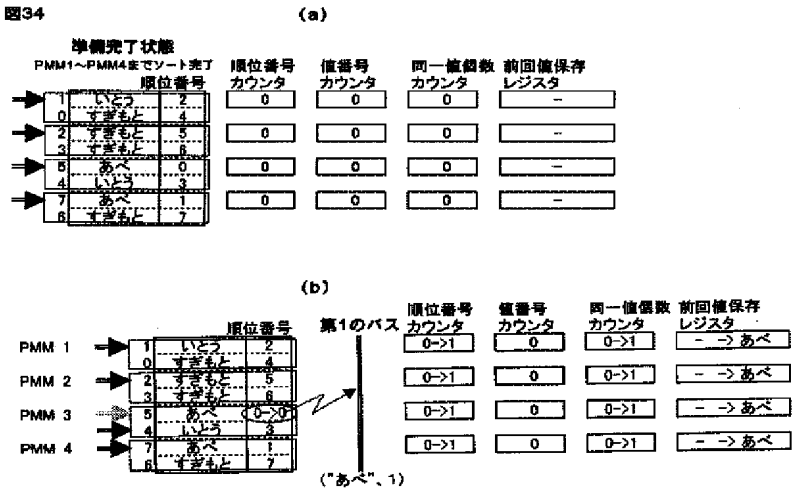


図34
(a)
準備完了状態
PMM1～PMM4までソート完了
順位番号
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
(b)
順位番号
第1のパス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
PMM 1
PMM 2
PMM 3
PMM 4
("あべ"、1)

【図３５】

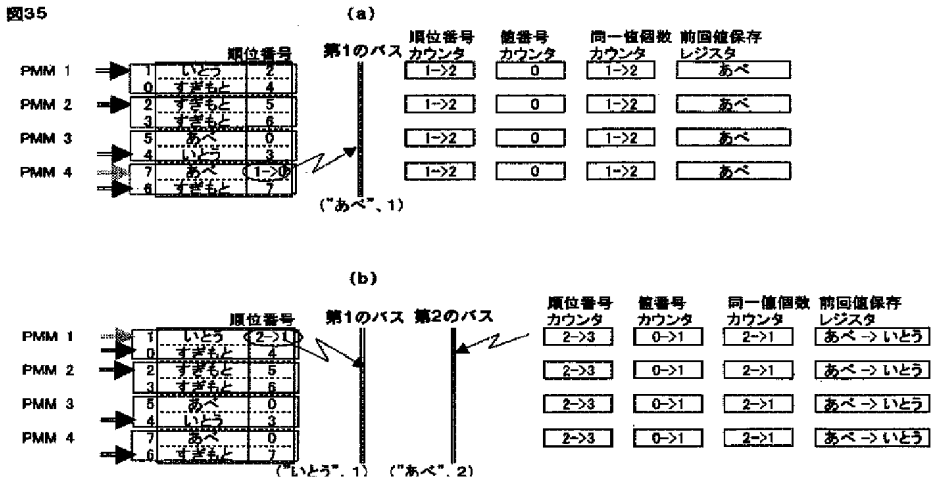


図35
(a)
順位番号
第1のパス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
PMM 1
PMM 2
PMM 3
PMM 4
("あべ"、1)
(b)
順位番号
第1のパス
第2のパス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
("いとう"、1)
("あべ"、2)

【図３６】

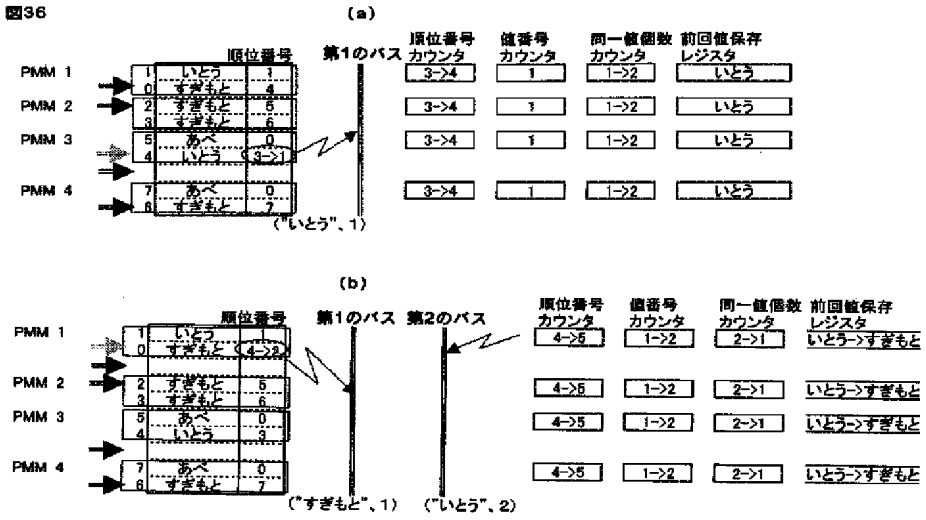


図36
(a)
順位番号
第1のパス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
PMM 1
PMM 2
PMM 3
PMM 4
("いとう"、1)
(b)
順位番号
第1のパス
第2のパス
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
("すぎもと"、1)
("いとう"、2)

【図３７】

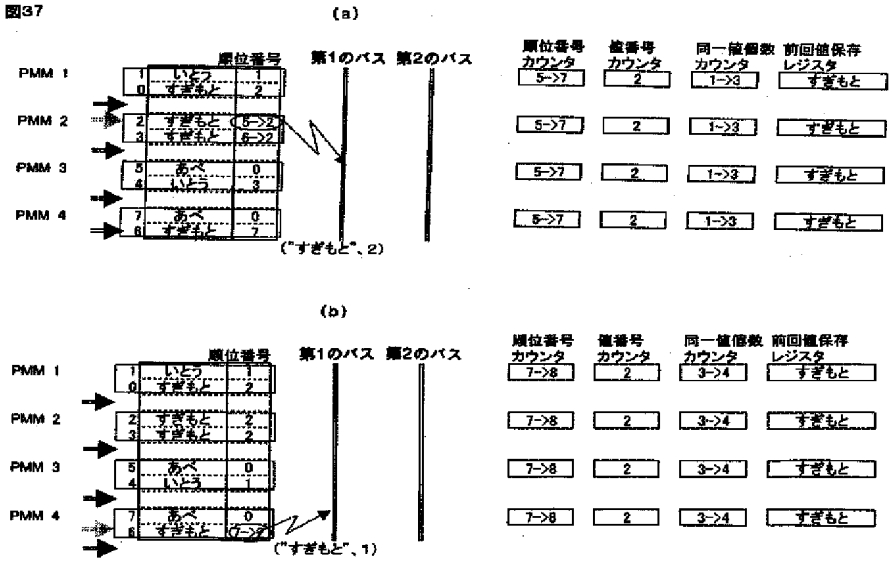


図37
(a)
順位番号
第1のパス
第2のパス
PMM 1
PMM 2
PMM 3
PMM 4
("すぎもと", 2)
順位番号カウンタ
値番号カウンタ
同一値個数カウンタ
前回値保存レジスタ
(b)
("すぎもと", 1)

【図４３】

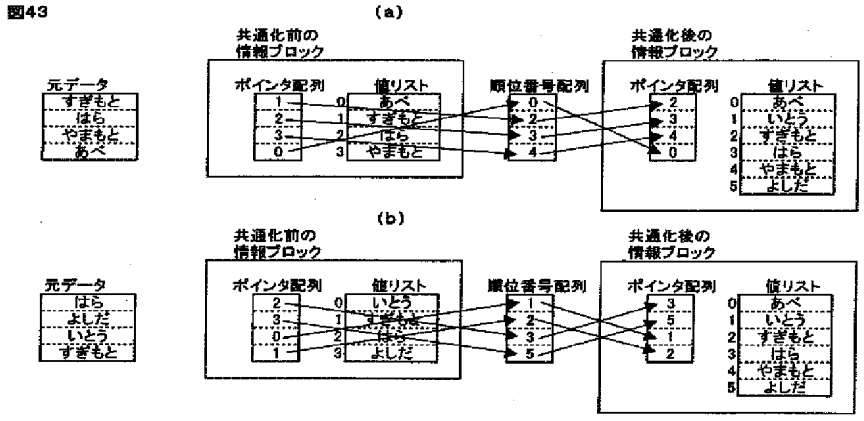
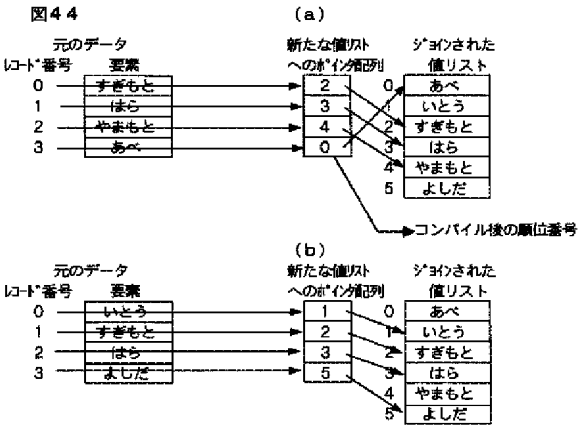


図43
(a)
共通化前の情報ブロック
共通化後の情報ブロック
元データ
ポインタ配列
値リスト
順位番号配列
(b)

【図４４】



フロントページの続き

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
		Ｇ０６Ｆ　15/403	３４０Ｄ